ABSTRACT

Title of Dissertation: IMPLEMENTING UNIVERSAL DESIGN TO SUPPORT NEURODIVERGENT STUDENTS IN UNDERGRADUATE INTRODUCTORY COMPUTER SCIENCE CLASSES

Emily Mae Kaplitz,
Doctor of Philosophy in Computer Science 2024

Dissertation directed by: Associate Professor, David Weintrop, and EDUC-Teaching and Learning, Policy and Leadership

There has been an increase of both neurodivergent students and enrollment in Computer Science programs in higher education. These increases have brought attention to two separate challenges: neurodivergent college students struggle more compared to their neurotypical peers and many students struggle in introductory computer science courses. This study considers both of these realities by investigating the impact of using Universal Design for Learning (UDL) strategies to revise introductory computer science courses. In doing so, it seeks to understand if and how UDL strategies can support neurodivergent students, along with their neurotypical peers, in succeeding in historically challenging academic contexts.

This study focused on three questions: How can universal design for learning principles be implemented into an introductory programming course?, How does updating an introductory programming course to follow the universal design for learning framework affect the experience of students?, and What are the experiences of neurodivergent and neurotypical students after updating an introductory programming course to follow the universal design for learning framework?

This study was conducted over three semesters of introductory programming courses. The first semester served as a control for the study where no changes were made to the course. In the following semester, the course's lecture slides were updated to be more accessible following Universal Design for Learning principles. In the third and final semester, the course's lecture slides were updated to be more accessible, and supplementary slides were created for the projects to help students understand what is being asked of them. To assess the interventions, student surveys, interviews, and grades were analyzed.

The findings serve as a demonstration of how to modify lecture slides and create programming project slides to apply universal design for learning principles and show the students had considerably better experiences and greater academic success in the course that applied universal design for learning principles. These benefits were recorded for both neurodivergent and neurotypical students, showing that universal design for learning principles benefits all students. The implications of these findings are considered and recommendations for improving introductory programming instruction are discussed. This work advances our understanding of how to help students comprehend foundational computer science concepts and develop the necessary computer science practices to excel in the field.

IMPLEMENTING UNIVERSAL DESIGN TO SUPPORT NEURODIVERGENT STUDENTS IN UNDERGRADUATE INTRODUCTORY COMPUTER SCIENCE CLASSES

by

Emily Mae Kaplitz

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science
2024

Advisory Committee:

David Weintrop

William Gasarch

Zhicheng (Leo) Liu

Caro Williams-Pierce

Gulnoza Yakubova

David Mount

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ADHD - Attention Deficit Hyperactivity Disorder

AP - Advanced Placement

ASD - Autism Spectrum Disorder

CMSC 131 - Object-Oriented Programming I

CMSC 132 - Object-Oriented Programming II

CS - Computer Science

DSM-IV-TR - Diagnostic and Statistical Manual of Mental Disorders 4 Text Revised

DSM-V - Diagnostic and Statistical Manual of Mental Disorders 5

IDE - Integrated Development Environment

RAADS-R - Ritvo Autism Asperger Diagnostic Scale–Revised

TA - Teaching Assistant

UDL - Universal Design for Learning

UMD - University of Maryland

WCAG - Web Content Accessibility Guidelines

Chapter 1: Introduction and Motivation

Overtime, there has been an increase of Neurodivergent students pursuing higher education and Computer Science majors (Aguar et al., 2016; Clouder et al., 2020). This increase has brought attention to two problems: Neurodivergent college students face many struggles compared to their neurotypical peers and Computer Science is difficult for students to learn (Clouder et al., 2020; Medeiros et al., 2019). When considering both of these problems, a natural question emerges: *How can we improve the way we teach computer science courses to help students learn the material while also minimizing the struggles of neurodivergent students*? This dissertation aims to gather insight on possible solutions to the challenge by using Universal Design for Learning (UDL) to create a better introductory programming learning environment for both neurodivergent and neurotypical students.

When neurodivergent students transition from high school to university, they tend to receive less support than when they were in high school (Scott et al., 2003). Universities are not required to have specialized curricula or programs designed for neurodiverse students (Rao & Gartin, 2003). However, colleges do have to provide equal opportunities, access, and protection from discrimination. Colleges are legally bound to provide appropriate accommodations to students with a documented disability and who are registered with the college's disability services by the Americans with Disabilities Act (W. M. Hadley, 2007). Appropriate accommodations can include changes in the classroom environment, modifications to class policies, use of services, or modifications to programs. However, accommodations do not lower program standards (Rao & Gartin, 2003), but instead are designed to create an equal environment for disabled students and do not give an unfair advantage to these students. However, even with accommodations, disabled students have a lower graduation rate with only about 41% of students with disabilities graduating (White et al., 2016). Neurodivergent students often drop out of higher education

due to expectation on how to behave, study, and interact, negative perceptions of education, and reliance on traditional methods of learning and assessments (Hamilton & Petty, 2023).

Research done on neurodiverse college students is fairly scarce. Stigma related to neurodivergencies has led to a history of research being focused on how to "fix" the neurotype. This research tends to follow the medical model of disability (Dwyer, 2022) which views disability as a result from physical or mental health conditions which can be treated or cured through medical professionals (Hanif et al., 2024). This model of disability can lead to disabled people believing they can't participate in activities that nondisabled people can (Hanif et al., 2024). In contrast, the social model of disability is the idea that disability is due to society's attitudes and actions, rather than a condition itself (Goering, 2015). Neither model can be viewed by itself if one would like to understand disability. Only considering the medical model neglects social effects on a disabled person's experience (Hanif et al., 2024). Only considering the social model neglects the effects of symptoms on a person's experience (Hanif et al., 2024). The disabled experience often includes both medical and social barriers.

Consider a student with ADHD. This student may have experienced heavy stigma from others. They might have been labeled as lazy and irresponsible throughout their life. However after being diagnosed, they might start taking medication to help them with everyday tasks. The student still has ADHD even when on medication. They might still need accommodations as medication often doesn't 'fix' all the issues that the student might be having. Instead, it might help to decrease symptoms enough to allow the student to more easily complete tasks that previously would be challenging. Oftentimes those with ADHD forget to take their medication because of their ADHD, paradoxical to the belief that people with ADHD are addicted to their medication (Advokat & Scheithauer, 2013). Hence, a

combination of societal changes to minimize obstacles and diagnosis and treatment is needed in order for neurodiverse people to thrive. Therefore there needs to be research focused on the full nuances of neurodiversity instead of following a single model.

Research about neurodiverse students can be difficult when trying to study students that have chronic mental health issues. At any given time, there will be 10 to 20% of the student population that struggle with mental health (Bhujade, 2017). However, not all of those students that struggle with mental health are neurodivergent. Mental health issues can be acute or chronic. Chronic mental health diagnoses are neurodivergencies, while acute ones are not. Acute mental health issues often do not need treatment, unless there is an underlying medical problem. Chronic mental health diagnoses will last for years and often need to be treated by professionals. In research, it is often not clear if students are struggling with a mental health disorder or suffering from mental health issues due to the stress of higher education and adult life.

It has only been in recent years that research has been focussing on how to work with neurodivergent students instead of strictly changing them. There is a need for more research on neurodivergent students in higher education without looking at neurodiversity as something that needs to be fixed. There are neurodivergent people across economic status, race, sexual orientation, and gender. However, with this intersectionality comes the drawback of women and minorities experiencing issues when getting diagnosed (Demer, 2018; Kreiser & White, 2014). Past research on the neurodivergent population has focused on the behaviors and symptoms of relatively rich white male children (Mallipeddi & VanDaalen, 2021). A reason for this is that parents had the means to get these children help when they started to struggle in school or work environments. Further, neurodivergencies tend to present differently between those assigned to be female and male at birth (Kreiser & White, 2014). Therefore, it is important to keep in mind that

statistics on these disorders are likely to undercount the true number of people with these conditions.

There is not an estimation on how many people across the US are neurodivergent; however, it is estimated that 30-40% of the global population is neurodivergent (*Neurodiversity and Other Conditions*, n.d.). Note here that this statistic is often cited, but it is not clear where this estimation comes from. Neurodivergent conditions include Autism, Attention Deficit Hyperactivity Disorder, Dyslexia, Dyscalculia, Dyspraxia, Tourette Syndrome, chronic mental health issues, and other conditions that affect the brain. Autism Spectrum Disorder affects communication and behavior. Dyspraxia affects fine and gross motor skills, motor planning, and coordination. Dyscalculia affects learning or comprehending arithmetic. Dyslexia affects areas of the brain that process language. ADHD causes attention issues, hyperactivity, and impulsiveness. While Dyslexia and Dyscalculia are still terms that are used, the updated terminology for both conditions is a specific learning disorder (American Psychiatric Association, 2022). Tourette Syndrome causes repetitive movements or unwanted sounds.

Neurodivergencies often have comorbidities with other neurodivergencies. Often neurodivergent individuals have multiple neurodivergent conditions and other medical conditions. Those with ADHD are more likely to have a mood, conduct or substance disorder, suicidal ideation, autism, anxiety disorders, major depressive disorder, and specific learning disorders (American Psychiatric Association, 2022). Those with anxiety disorders are more likely to have suicidal ideation, depressive disorders and autism spectrum disorder (American Psychiatric Association, 2022). Autistics are more likely to have anxiety, depression, suicidal ideation, a specific learning disability, and ADHD (American Psychiatric Association, 2022). Those with a depressive disorder are more likely to have an anxiety disorder, eating disorder, alcohol and substance abuse, and medical

conditions (American Psychiatric Association, 2022). Those with specific learning disorders are more likely to develop anxiety, depression, behavioral problems, suicidal ideation, ADHD, and autism (American Psychiatric Association, 2022).

These comorbidities tend to be diagnosed in adulthood rather than in childhood when a neurodivergence is first recognized. A child might only be diagnosed with the condition whose symptoms are clear in home and school environments. Thus, a child might not realize until adulthood that all their symptoms might not only be explained by the one diagnosis they received.

Around 9.4% of children have ADHD (CDC, 2020b). 1 in 54 children have been diagnosed with Autism Spectrum Disorder with some states having even higher rates (CDC, 2020a). Around 33% of children who receive accommodations in school have a specific learning disability (*Students With Disabilities*, 2023) with around 5-9% of the general population having a specific learning disability (Boat et al., 2015). Specific learning disabilities include dyscalculia and dyslexia. About 10% of the population has mild to severe cases of dyspraxia (Gibbs et al., 2007). It is estimated that 0.6% of children have Tourette Syndrome (CDC, 2021).

In 2008, approximately 11% of postsecondary students in the US had a disability. Of that 11%, less than 10% of students had a specific learning disability (Pino & Mortari, 2014). About 25% of college students receiving disability services have ADHD. It is estimated that 2-8% of college students in the US have ADHD (Green & Rabiner, 2012). Autistics are less likely to enroll in a university than those with other disabilities (White et al., 2016).

More statistics on the neurodivergent population are not known. Part of the reason is that 2 to 50% of a student population will struggle with stress, anxiety, and depression

(Bhujade, 2017). The statistics on neurodiversity can also often be skewed and unreliable due to underreporting and possible underdiagnosing that is a result of too narrow of definitions of diagnoses (Clouder et al., 2020). Another reason is that neurodivergencies are still very hidden from the public eye, so unless you have dyslexia, ADHD, or ASD, the general public often does not know about other conditions. However, even though the general public has a broad understanding of dyslexia, ADHD, or ASD the understanding of the conditions are coming from stereotypes (Draaisma, 2009). This affects what researchers study. So, there is a need for more work to be done on the experience of neurodivergent students at institutions of higher education. Without this work, we will continue to fail neurodivergent students with lack of resources and empathy for their struggles.

A lack of support for neurodivergent students in higher education has resulted in consequences. For example, students with disabilities are underrepresented in STEM fields (Chrysochoou et al., 2022). This might be due to struggles in learning spaces designed for neurotypical students, limited exposure to prerequisite courses, a lack of role models, or a lack of access to individualized support (Dunn et al., 2012). Large universities have significant barriers that STEM undergraduates must overcome including large class sizes, fast paced instruction, a lack of support, and unreasonable standards are inaccessible (Schreffler et al., 2019). Therefore, there is a need for research on how to retain students with disabilities in STEM majors (Chrysochoou et al., 2022).

Computer science skills are important across STEM fields and also in many non-STEM related fields (Koushik & Kane, 2019). There is a growing demand for computer science skills. There are many benefits, especially for neurodivergent people, to learning computer science including employment opportunities, general improvements in problem solving, understandment of data, and developing technical skills (Koushik & Kane, 2019). However, programming courses tend to have the highest dropout rates and are regarded as difficult

(Robins et al., 2003). To help combat these dropout rates, programming courses need to be accessible to as many students as possible (Hansen et al., 2016). Research done in computer science education for disabled students tends to focus on accessibility of blind and visually impaired students (Koushik & Kane, 2019). Therefore, there is a need to better understand ways to make computer science accessible to the neurodivergent community.

This study examines how to make introductory programming courses more supportive to neurodivergent students. In particular, this study focuses on students with ADHD, ASD, Dyslexia, Depression, and Anxiety. This dissertation seeks to answer the following questions:

- How can universal design for learning principles be implemented into an introductory programming course's materials?
- How does updating an introductory programming course to follow the universal design for learning framework affect the experience of students?
- What are the experiences of neurodivergent and neurotypical students after updating an introductory programming course to follow the universal design for learning framework?

The study was conducted during three separate semesters with the same instructor. The first semester was the control group. The students were tasked with taking surveys, but no changes to class material were made. The second semester received material that was deemed accessible by accessibility standards. The third semester's students received accessible material along with animations to go with their projects that were tailored to be helpful to neurodiverse students.

The remainder of this dissertation consists of 5 chapters. The next chapter, chapter

2, presents a review of literature relevant to the study. This includes an explanation of Neurodiverse conditions, Neurodivergent college students experiences, the Universal Design for Learning Framework, and current ways of teaching programming to college students. Chapter 3 describes the study design and methodological approach used to answer the research questions and discusses the settings and participants. It also outlines the data collection strategies employed and analytic approach used. Chapter 4 assesses the course before any design changes. It then explains how the materials were updated and how new material was designed. Chapter 5 presents the research findings and how those findings answer the focal research questions and their subquestions. The dissertation concludes with a discussion of the implications of this work, conclusions that can be made from it, and the limitations of the study.

Chapter 2: Literature Review

This literature review is split up into four different sections: Neurodiversity, Neurodivergent College Students, UDL (Universal Design for Learning) Framework, and Teaching Programming. The Neurodiversity section starts by defining neurodiversity. It further explains the different neurodivergent conditions and how they affect a person's life. It concludes with an explanation of how different neurodiverse conditions can interact with each other when someone is diagnosed with more than one condition. The Experiences of Neurodivergent Undergraduate Students section explains what college life is life for a neurodivergent college student. It starts by discussing how disability services work. Then it explains some struggles that neurodivergent college students face and how to best teach neurodivergent students. It concludes by discussing the attitudes that instructors can have about neurodiversity. The UDL Framework starts by defining Universal Design. It then discusses what the Universal Design for Learning Framework is. It concludes by discussing the UDL efficacy for different types of students. Finally, the Teaching Programming section presents research on the difficulty of learning to program. It concludes with a discussion of UDL in introductory programming contexts and research on teaching programming to neurodivergent students.

2.1 Neurodiversity

The neurodiversity movement is an extension of the disability rights movement (Mcgee, 2012). The term neurodiversity was created by a sociologist named Judy Singer in 1999 (Doyle, 2020). She used the term to help explain her experience of being an autistic woman. Neurodiversity was a term meant to be used similar to biodiversity: All brains are unique and these diversities are essential (Dwyer, 2022). Thus, neurodiversity would refer to the differences in brain function and behavior in a population (Clouder et al., 2020). It

challenges the idea that conditions that result in different brain function and behavioral traits are only negative (Clouder et al., 2020).

Brain variation in humans is as important as normal biodiversity is to the ecosystem (Mcgee, 2012). This variation allows for humans to have a vast amount of skills which helps society at large. Consider a student project that involves creating a video game. Creating a video game has many different parts including writing a story, coding the game, designing the artwork, beta testing, ect. All four of the parts mentioned need different skill sets. It is very rare that a single person would be able to have all skills needed for the project.

However, the term changed to be a contrast to those that have 'normal' brains. Here 'normal' means to not have a cognitive disability or chronic mental illness. Neurodiversity advocates for understanding that autism and other cognitive disorders are not diseases that need to be cured, but are vital to society. Neurodivergent became a term used to describe a person who has a cognitive disability such as Autism, Attention Deficit Hyperactivity Disorder, Dyslexia, Dyscalculia, Dyspraxia, and Tourette Syndrome (Clouder et al., 2020). This term continues to evolve to describe other conditions like chronic mental health conditions. It is not used to exclude anyone who does not fit into what society thinks is 'normal' cognitive function. In contrast, Neurotypical is used to describe anyone who is not neurodiverse.

The most common neurodivergent conditions are attention deficit hyperactivity disorder (ADHD), autism spectrum disorder (ASD), depressive disorders, dyslexia, and anxiety disorders. Due to the prevalence of these conditions, this dissertation uses these five conditions as a basis when considering how we can help neurodivergent students. Focusing on only five conditions allowed for more clarity on the needs of these students. ADHD causes attention issues, hyperactivity, impulsiveness, and executive dysfunction. ASD affects communication and behavior. Depressive disorders cause a depressed mood and loss

of interest in activities that affect everyday life. Dyslexia affects areas of the brain that process language. Anxiety disorders cause immense worry, anxiety, and fear on a regular basis. In this dissertation the term neurodiverse refers to the five most common neurodiverse conditions: ADHD, ASD, dyslexia, depressive disorders, and anxiety disorders. The following sections will discuss these conditions in more detail. For each disorder, symptoms will be discussed as well as how the condition might impact students in the classroom.

2.1.1 Attention Deficit Hyperactivity Disorder

Attention Deficit Hyperactivity Disorder (ADHD) involves a pattern of inattention, hyperactivity, and impulsivity that interferes with everyday life (American Psychiatric Association, 2022). Informally, inattention is the inability to complete a task that is not caused by the person not wanting to do the task. Hyperactivity refers to movement at inappropriate times. Impulsivity is the inability to think through consequences of actions which can cause harm to the individual or another person. Impulsivity may be caused by the desire for rewards and the inability to wait for something that is craved (American Psychiatric Association, 2022).

There are three types of ADHD: Inattention, Hyperactivity, and Combination (Both Inattention and Hyperactivity). If a person is diagnosed with Inattention or Hyperactivity it does not mean they will not struggle with any of the other symptoms presented in the other type of ADHD. ADHD often presents as a behavioral issue. Those with ADHD will often have problems with impulsivity and emotions. This may present as a person who is quick to anger, frustration, and other emotions that might not be appropriate for a given setting (American Psychiatric Association, 2022). Often students with ADHD will struggle with school more than their neurotypical peers even though ADHD is not a learning disability. Therefore, ADHD is still a disability, but a disability that does not only affect a person's

intellectual ability to do school work. People with ADHD are often seen as lazy, irresponsible, and hard to work with (American Psychiatric Association, 2022).

Someone who has Inattentive ADHD might struggle with paying close attention to detail, staying on task, being perceived as distracted, following directions, completing tasks, organizing tasks, avoiding tasks that are not interesting, losing things, distractedness, and forgetfulness (American Psychiatric Association, 2022). In a classroom setting, Inattentive ADHD can look like making small easy to fix mistakes on an assignment, spacing out during lectures, not doing the work assigned in class, disorganized assignments, missing due dates, handing in assignments that are rushed or incomplete, not handing in assignments, coming to class unprepared, forgetting about quiz or exam dates.

Someone who has Hyperactive ADHD might struggle with fidgeting, sitting for long periods of time, feeling restless, engaging in activities quietly, talking excessively, waiting for people to finish speaking, waiting in lines, and interrupting others (American Psychiatric Association, 2022). In a classroom setting, Hyperactive ADHD might look like a student asking to use the bathroom, stimming, interrupting the instructor or other students, and asking unrelated questions. Stimming refers to specific behaviors that include hand-flapping, rocking, spinning, repetition of words and phrases, playing with rings, playing with hair, or tapping pens and may appear differently across Neurodivergent conditions. However, you do not have to be neurodivergent to stim. Stimming is a self-soothing tool that can help a person with ADHD to stay on task.

Students with ADHD often struggle with their emotions (DuPaul et al., 2017). This might cause these students to have difficulties keeping friends. Which as a result causes the student to feel more isolated. Students with ADHD are more at likely to participate in nonacademic related behaviors (Meaux et al., 2009). This could be as innocent as spending too much time playing video games or as dangerous as participating in substance abuse.

This is due to the ADHD brain seeking out quick dopamine hits that their brain is often lacking. Either behavior can result in poor academic performance. Students that struggle with executive function, like those with ADHD, struggle with being on campus to study. Often they will get distracted by others and not be able to focus on tasks (Van Hees et al., 2015). Students with ADHD will often have anxiety about what they will do after college due to accommodations for ADHD being not as available in the real world (Van Hees et al., 2015).

ADHD presents differently between sexes. When looking at those with ADHD, people assigned male at birth tend to show it outwardly because they are more often diagnosed with the hyperactivity type (Nussbaum, 2012). Those assigned female at birth tend to have an inattentive type, so their symptoms present internally (Nussbaum, 2012). Therefore, it is likely that the number of people assigned female at birth are underdiagnosed. Currently, ADHD is considered more frequent in those assigned male at birth than those assigned female at birth with a ratio of 2:1 in children and 1.6:1 in adults (American Psychiatric Association, 2022).

2.1.2 Anxiety Disorders

There are many different anxiety disorders including separation anxiety disorder, social anxiety disorder, panic disorder, generalized anxiety disorder, substance/medication-induced anxiety disorder, anxiety disorder due to another medical condition, ect (American Psychiatric Association, 2022). All anxiety disorders share symptoms of excessive fear and anxiety. Fear refers to an emotional response to a real or perceived threat, while anxiety is the anticipation of a threat (American Psychiatric Association, 2022). Panic attacks are a common symptom of anxiety disorders. The difference between anxiety disorders are the situations in which that disorder brings on fear and anxiety (American Psychiatric Association, 2022). This means that anxiety disorders can be comorbid to each other.

Symptoms of the anxiety disorder can be brought on by stress and environment. Anxiety disorders often present in childhood and persist throughout adulthood (American Psychiatric Association, 2022). Anxiety disorders are twice as prevalent in those that are assigned female at birth than those assigned male at birth (American Psychiatric Association, 2022).

When anxiety disorders are left untreated, they can persist over extended periods of time. This may impact every part of a person's life and can be extremely debilitating in the worst cases. In the classroom, a student who is struggling with anxiety can look like a person who keeps to themselves, does not raise their hand to ask or answer questions, and hands in work that is incorrect that could have been clarified before the assignment was due (Jones et al., 2019). Contradictory, it can also look like a student who asks too many questions, asks the instructor to repeat previously stated instructions, and a student that is constantly in office hours in order to make sure their work is correct. Students that struggle with anxiety may also only struggle with certain parts of a course. For example, a student might really struggle to control their anxiety during an exam, but be perfectly fine with presenting a project and vice versa. A student's behavior might also be affected by their anxiety. This could look like a student constantly fidgeting, avoiding a specific task, being irritable, or failing to complete a task (Killu et al., 2016). Anxiety also can affect a student physically. A student who is struggling with anxiety might also have tics, heart rate increase, headaches, nausea, and muscle tension (Killu et al., 2016).

2.1.3 Autism Spectrum Disorder

Autism Spectrum Disorder (ASD) is a neurodevelopmental disorder that a person would have since birth. However, a person might not be recognized as having Autism until later in life as symptoms can vary and individuals might not outwardly present impairments in early childhood due to their environment masking the issues (American Psychiatric Association, 2022). Those with ASD struggle with deficits in social communication and

interaction and restricted and repetitive patterns of behavior (American Psychiatric Association, 2022). Deficits in social communication and interaction may look like abnormal social interaction, nonverbal communication behaviors, and difficulties adjusting to behaviors expected from them in different social contexts (American Psychiatric Association, 2022). In the classroom, this might look like a student struggling to work in a group of other students, not looking at a lecturer or another student as they speak, and struggling to make friends.

Restricted and repetitive behaviors may look like repetitive movements or speech, inflexible routines, fixated interests, and a sensitivity or lack of sensitivity to sensory input (American Psychiatric Association, 2022). In the classroom, this could look like a student clicking their pen excessively even during times when the student is meant to be quiet, becoming distressed when a due date is changed, asking about only one topic of the subject, or having an adverse reaction to changes in the physical classroom environment. Fixated interests are also known as special interests. Special interests can be a source of motivation to pursue education and employment in a field that relates to their interest (American Psychiatric Association, 2022).

Autistic without cognitive or language impairments may be able to outwardly present as allistic. Allistic is a term that means a person who is not on the Autism Spectrum. A person’s social communication issues might be subtle if the person has overall good communication (American Psychiatric Association, 2022). This can happen if a person doesn’t have intellectual impairments, so they are able to create meaningful sentences and be an active member in a conversation. A person's restricted and repetitive behaviors may not be obvious if the interests are closer to what society would call normal (American Psychiatric Association, 2022). For example, a child would not be considered acting inappropriately if they continuously chewed on their pencil or rocked back and forth on a

desk chair with wheels. The child's actions would be looked at as a bad habit instead of something that could be a larger issue. By the time the child becomes an adult, other people might see the behavior as childish, but ultimately it would also similarly be looked at as only a bad habit. Autistic adults are often not perceived as strange if they pursue jobs and only want to talk about their job that relates to their special interest (American Psychiatric Association, 2022).

Autistics also can have an uneven level of skill, difficulties seeing the world from another person's perspective, executive function deficits, and focussing on details rather than a larger scope (American Psychiatric Association, 2022). In the classroom, this can look like a student having excellent work when working individually and poor work when forced to work with otters, difficulties trying to figure out how someone else solved a similar problem, not being able to make themselves do a certain tasks and able to do other tasks easily, and asking an instructor to go in depth on a topic when the course only requires a high level overview.

Autistic students will often struggle with adjusting to college. This often results in anxiety and stress, loneliness, loss of structure, and sensitivity to the new environment (Clouder et al., 2020). Often the student will not be able to cope and will cause poor academic performances. Autistic students will often have anxiety about future prospects due to not having the familiar structure of school and difficulties they can face in the workforce (Bolourian et al., 2018). Due to autistic students struggling with social cues and other social expectations, students will often not initiate social interactions with others. Autistic students will often struggle with social anxiety and overstimulating environments, so they will often keep to themselves in quiet environments where they are less likely to meet people (Vincent et al., 2016). Autistic students will often struggle with time-management and organization (White et al., 2016). They will often have issues with identifying important

information in a course as they tend to focus on details rather than the broad concept and can struggle with information processing (Clouder et al., 2020).

Autism can appear differently in those that are assigned male at birth vs female. Those that are assigned male at birth are three to four times as likely to be diagnosed then those assigned female at birth (American Psychiatric Association, 2022). However there are concerns that this ratio underrepresents the amount of assigned female at birth autistics. The average age of diagnosis in those assigned female at birth is later than those assigned male at birth (American Psychiatric Association, 2022). Those that are assigned female at birth tend to have a comorbid disorder as well. Those without comorbid disorders often might not get a diagnosis of ASD(American Psychiatric Association, 2022). This could mean that those assigned female at birth might present more subtly, so they can often cope easier than those assigned male at birth.

2.1.4 Depressive Disorders

There are many different depressive disorders including disruptive mood dysregulation disorder, major depressive disorder, persistent depressive disorder, substance/medication-induced depressive disorder, ect (American Psychiatric Association, 2022). However, all depressive disorders share persistent symptoms of sad and empty or irritable mood that can impede a person's life (American Psychiatric Association, 2022). The difference between the disorders relate to the duration, timing, and cause of the episode (American Psychiatric Association, 2022). It's important to note that depressive disorders are a chronic condition. Acute sadness and empty or irritable mood due to an event like grief after a death is not a depressive disorder. Depressive disorders can be diagnosed in both childhood and adulthood.

Depressive disorders can persist over a long period of time. Severe depression can affect all aspects of a person's life leading to a destructive cycle that is incredibly hard to escape (American Psychiatric Association, 2022). In the classroom, a student who is struggling with a depressive disorder can look like someone who rarely comes to class and when they do come to class they are not engaged (*Understanding AND Accommodating Students with Depression IN THE Classroom*, n.d.). It can look like a student missing assignments and never reaching out to the instructor for help. Conflictingly, a student can look and act completely "normal". They come to class, never miss assignments or deadlines, and engage in the course. However, they might not have a healthy social life, not be eating or not eating healthy, and they might either be sleeping too much or not enough (*Understanding AND Accommodating Students with Depression IN THE Classroom*, n.d.). A student that has struggled with depression for a long time might know how to cope, but they might not be coping healthy. A depressed student might easily become irritable and be "fine" one day and make self-destructive choices the next (*Understanding AND Accommodating Students with Depression IN THE Classroom*, n.d.).

2.1.5 Dyslexia

Dyslexia is a term used to refer to a pattern of problems with word recognition and poor spelling abilities. A dyslexic may struggle with reading accuracy, rate or fluency, or comprehension (American Psychiatric Association, 2022). Often a person diagnosed with dyslexia would also struggle with spelling accuracy, grammar and punctuation accuracy, and clarity or organization of written expressions (American Psychiatric Association, 2022). However, in recent years, dyslexia is no longer used as a diagnostic term. Instead, Specific Learning Disorder is used as it encompasses more symptoms and specifics (American Psychiatric Association, 2022). I have chosen to continue to use the term dyslexia as it is more known to the general public and the specific learning disorder I am referring to in this

paper is strictly about learning difficulties related to language processing. This would include both impairments with reading and writing.

A specific learning disorder is a neurodevelopmental disorder that is present at birth. The most common form of specific learning disorder is what the general population refers to as dyslexia, or the difficulty of mapping letters to sound in order to read (American Psychiatric Association, 2022). In order for a person to be diagnosed with a specific learning disorder they must have difficulties learning and using academic skills (American Psychiatric Association, 2022). This may present as inaccurate, slow, or difficulties in word recognition, difficulties understanding what is written, difficulties with spelling, difficulties with written expressions, difficulties with numbers and calculations, or difficulties with mathematical reasoning (American Psychiatric Association, 2022). In the classroom setting, this can look like a student taking more time to complete a written task than others, not following instructions as they do not understand what is written, handing in work that often has many grammar or spelling errors, not being able to do simple addition, subtraction, multiplication, or division but able to do more complicated procedures, struggling with math concepts when first presented.

A person with a specific learning disorder may struggle with academic skills that are below what is expected for the person's age (American Psychiatric Association, 2022). Often difficulties will present early in a person's childhood but not be fully recognized until the person can no longer cope with academic demands. A specific learning disorder doesn't just affect one part of a person’s life. In the academic space difficulties in one skill whether it be reading, writing, or math can bleed into other subjects making those subjects harder for an individual (American Psychiatric Association, 2022). For example, a student that was diagnosed with a specific learning disorder that affects their ability to read and write or what was traditionally known as dyslexia, might be naturally gifted at mathematics compared to

their peers. However, on an algebra quiz there is a question that is worth 15 points that is a word problem. Therefore, even though the student could easily solve every other question, they would continuously only receive an 85 on quizzes as they could not comprehend what the question was asking of them. Learning difficulties are not due to an intellectual, visual, or auditory disability nor can the difficulty be explained by other mental or neurological disorders (American Psychiatric Association, 2022).

Dyslexic students often struggle with following lectures. This often leaves students being easily distracted and have poor notes (Nightingale et al., 2019). Students with dyslexia will often struggle with interpreting what is important and what is not important due to struggles with comprehension. Students with dyslexia often have problems with short-term memory (Nightingale et al., 2019). So, they might experience anxiety and embarrassment about asking questions because they are worried that the instructor will get annoyed about having to repeat themselves. Due to the nature of specific learning disabilities, students with dyslexia will often struggle with college expectations (Glazzard & Dale, 2015). College instructors often expect a high level of reading, writing, and grammar skills (Clouder et al., 2020). A dyslexic student can often cope with this using technology. However, many college courses require students to do work on paper, like exams. This puts dyslexic students at a huge disadvantage because they no longer have their technology to help correct their grammar and spelling. A student with dyslexia may struggle with reading, but that doesn't mean they are completely illiterate. Often a student with dyslexia has a large vocabulary, but they struggle with word recognition and recall. Thus, when a student is writing a paper they may choose to use simple language rather than the larger extent of their vocabulary.

There are different levels of severity of dyslexia that students can present with (American Psychiatric Association, 2022). They can show mild symptoms which include

difficulties learning skills in less than two subjects, but the student is able to cope with accommodations or support services. Moderate symptoms present as a student struggling in more subjects and unlikely to succeed without specialized teaching and support. Someone with a severe specific learning disorder would struggle in many academic areas and would not likely be able to cope without individualized and specialized teaching. Severe symptoms can make it so that a student may never be able to do certain tasks efficiently even with significant accommodations. Having a specific learning disorder can cause lower academic achievement, higher rates of dropout of high school, lower rates of receiving a postsecondary education, a worse overall mental health, higher rates of unemployment, and a lower income (American Psychiatric Association, 2022).

2.1.6 Comorbitities

Those with one neurodiverse condition are more likely to have other neurodiverse and medical conditions. This makes navigating the world more difficult, especially as individuals will usually only be diagnosed with one condition. This also makes diagnoses of more than one condition more difficult because neurodivergent conditions often share symptoms with each other (American Psychiatric Association, 2022). Stricter diagnosis criteria in updated versions of the Diagnostic and Statistical Manual of Mental Disorders (DSM) has made it easier for diagnoses to be separated from each other.

Those with ADHD have a higher risk of suicidal ideation and behavior in children (American Psychiatric Association, 2022). In adults those with ADHD are at higher risk of attempting suicide and developing a mood, conduct or substance disorder (American Psychiatric Association, 2022). Those assigned female at birth have a higher rate of comorbid disorders including autism spectrum disorder and personality and substance use disorders. Anxiety disorders, major depressive disorder, and specific learning disorders also appear more in those with ADHD (American Psychiatric Association, 2022).

Those struggling with anxiety disorders are more likely to have suicidal ideation and commiting suicide (American Psychiatric Association, 2022). Those with anxiety disorders are also more likely to have comorbid depressive disorders and autism spectrum disorder (American Psychiatric Association, 2022).

Autistics with low support needs are prone to anxiety and depression (American Psychiatric Association, 2022). Autistics are at greater risk of self harm and committing suicide compared to allistics (American Psychiatric Association, 2022). Autistics have a greater chance of also having a specific learning disability and ADHD (American Psychiatric Association, 2022).

Someone struggling with a depressive disorder might also struggle with generalized anxiety disorder (American Psychiatric Association, 2022). Women are more likely to have comorbid anxiety disorder and eating disorders, while men are more likely to have comorbid alcohol and substance abuse (American Psychiatric Association, 2022). Someone with a depressive disorder may also have comorbid medical conditions.

Those with specific learning disorders are more likely to develop anxiety, depression, and behavioral problems. In the US and Canada, poor literacy and having a specific learning disorder is associated with suicidal thoughts and behaviors (American Psychiatric Association, 2022). A person who has one type of specific learning disorder often has different types. Those with a specific learning disorder also commonly have ADHD and ASD. Those with ADHD and a specific learning disorder have worse mental health than those with just a specific learning disorder (American Psychiatric Association, 2022).

2.2 Experiences of Neurodivergent Undergraduate Students

In recent years there has been an increase in the number of neurodivergent students attending higher education (Clouder et al., 2020). Institutes of higher education by law are

required to provide services to students with disabilities. These services are meant to assist with challenges associated with their specific disability and needs. However, these services require students to self disclose their disability to the college they attend and self advocate for any services they need. This means that a student needs to have a full understanding of their disability and be able to convince the college's disability services why an accommodation is necessary and why this accommodation is reasonable to be provided (W. M. Hadley, 2007).

Research on the experience of neurodiverse students in undergraduate programs reports that they often feel isolated, alone, stressed, anxious, unhappy, tired, depressed, and overwhelmed (Anderson et al., 2018). Experiences of neurodiverse students are greatly improved when reasonable adjustments are made (Hadjikakou & Hartas, 2008). However, students often will only disclose their disability when they can no longer cope or when they perceive it is safe to do so (Clouder et al., 2020). When disclosing learning difficulties, neurodivergent students can often experience poor treatment, a lack of support, inflexibility from lectures, perceptions of discrimination, and judgmental attitudes (Griffin & Pollak, 2009). These negative instructors' attitudes are detrimental to disabled students and will often result in the student experiencing more struggles (Glazzard & Dale, 2015).

2.2.1 Disability Services

Both public and private universities are required to have disability services to provide equal access to students with disabilities (ADA National Network, 2024). Universities are required to provide an accessible infrastructure, aids and services for communication, and have modification policies. Accommodations for neurodivergent students fall under the university's modification policies. Accommodations and modifications are required to be individually designed to make appropriate adjustments for each student's individual disability (ADA National Network, 2024). Disabilities present differently in each person,

therefore, there cannot be a standard accommodation list for any diagnosis. Since there are also comorbid conditions like a student can have both anxiety and dyslexia, each case must be considered separately. Accommodations and modifications are not required when providing the accommodation would fundamentally alter the course (ADA National Network, 2024). There is no clear definition for what would fundamentally alter a course. Therefore, an instructor can deny any accommodation. When a student is denied an accommodation, they would then have to talk to the disability services again to see if the instructor and disability services can come to an appropriate resolution. However, students are not guaranteed that the appropriate resolution would provide the accommodations needed.

Disability services are only granted to those that disclose their disability (Anderson et al., 2018). The student must go to the disability services with medical documentation of their disability inorder to be granted the ability to receive accommodations (W. M. Hadley, 2007). This documentation is often an assessment report from the professional that diagnosed the student with a particular disability and any accommodations students received previously. This document validates that the student does have a disability that requires accommodations (W. M. Hadley, 2007).

It is often hard for a student to get accommodations for a specific problem due to the school not being able to suggest accommodations for a student (Hsiao et al., 2018). Instead, the student must specifically ask for the accommodation. Often a student might know what they are struggling with but not know what they actually need to help the problem (Hsiao et al., 2018). For example a student might have severe anxiety, especially when registering for classes. However, it is not common knowledge that a student can ask for the accommodation of priority registration so the student doesn’t have to be worried about not getting the classes they need in order to graduate.

Common accommodations that help most neurodivergent students include extended time, distraction free environments, flexible due dates, ability to move exams so a student is not taking more than one exam a day, and flexible assignments (Clouder et al., 2020). These accommodations are often a necessity for neurodivergent students to succeed (W. M. Hadley, 2007). Accommodations are made so that a student's academic achievement reflects the actual knowledge and achievement that a student has rather than the obstacles created by their disability (Rao & Gartin, 2003).

2.2.2 Struggles Neurodivergent Students Face

Neurodivergent college students' issues are often not discussed or known by instructors (Birdwell & Bayley, 2022). This is partly due to the prevalence of neurodiversity in higher education being hard to determine because of a reliance on self-disclosure, diagnosis criteria changing, and condition specifics (Clouder et al., 2020). A student might not want to self-disclose their disabilities to instructors or the school due to fear of stigma. A student might be afraid of the instructor's reaction to the student having a disability. They might feel the instructor might assume the student wants “special treatment”. Neurodivergent students often experience a lack of support from lecturers who do not want to accommodate students. This is often a result of the lecturer feeling that an accommodation harms the integrity of the class. This leads students feeling like they are being treated unfairly and judged (Bolourian et al., 2018). Paradoxically, the student might worry that the instructor is giving “special treatment” to the student and therefore does not expect as much from the neurodiverse student compared to their neurotypical peers.

Instructors are often informed what accommodation a student needs, but are not told why the student needs the accommodation unless the student themselves discloses their disability to the instructor. While often not the intent of the lecturer, the lack of knowledge about neurodiversity and accommodations is detrimental to students (Clouder et

al., 2020). Generally, instructors are not equipped to support neurodivergent students. They rely on the disability services at the school to tell them what they need to provide (Clouder et al., 2020). In recent years, there has been a push to educate faculty about racial, cultural, and gender diversity (Ladner & Israel, 2016). However, there is an essential need for faculty to be educated about disability and in particular neurodiversity. This will lead to students feeling safer to disclose their disabilities to both the school and individual instructors.

Universities have the ability to minimize challenges that cause neurodiversity to be a disability (Clouder et al., 2020). Universities often offer support services like support groups, counseling services, and transition programs (Barnhill, 2016). However, these support services can be inconsistent (Griffin & Pollak, 2009). Generally, these services have a lack of funding and professionals that are equipped to deal with neurodiversity (Barnhill, 2016). This can often make support services ineffective. Since services are often limited and ineffective, students often find that the accommodations available to them are not sufficient to meet academic expectations (W. M. Hadley, 2007). Improving these support services is often difficult, time-consuming, and can be emotionally demanding on staff (W. M. Hadley, 2007).

Both neurodivergent and neurotypical students struggle with common stressors like academic demands, independence in a new environment, changes in family and social life, and exposure to new people (Bhujade, 2017). However, adjusting to college can be particularly difficult for neurodivergent students. There can be an overwhelming sense of being different and not knowing how to cope with the full ‘college experience’. Previous incidents in the students' life can lead to fear of judgment from both peers and instructors when entering college (Clouder et al., 2020). Creating an inclusive and trusting environment is essential.

Students with learning disabilities often have negative university experiences. These students often experience helplessness and hopelessness due to anxiety about a lack of understanding and inadequacy (S. C. K. Shaw & Anderson, 2018). These negative experiences can be exacerbated if the student does not have the necessary assistive technology and accommodations (W. Hadley, 2017). Neurodivergent students often feel isolated, stressed, anxious, depressed, and overwhelmed. This can lead to them isolating themselves from others as a result of past experiences or anxieties about having negative experiences while trying to make friends (Griffin & Pollak, 2009).

2.2.3 Technological Supports for Neurodivergent Students

The use of technology can be beneficial as students are able to utilize their tech based assistive technology. However, the use of technology is not always beneficial to students. Virtual learning is often a hindrance for neurodivergent students (Habib et al., 2012). Neurodivergent students struggle with traditional teaching and assessments. This causes stress when it comes to academic expectations like timed homework and quizzes, taking notes, and course load (Bolourian et al., 2018). Having only essays and written assessments can be detrimental to students. Neurodivergent students, particularly dyslexics, often benefit from equivalent forms of assessment like multiple choice and matching questions (Taylor et al., 2016).

Most college students are exposed to multiple types of technology. Among this technology, they may also be exposed to technology useful in a classroom setting. However, often students with disabilities are not aware of existing software or assist devices that can be helpful beyond the accommodations they are used to (Getzel, 2008). There is often an assumption that because the college students have computer skills and exposure to technology, they would also have access and knowledge about software like text-to-speech that can assist them with any academic challenges they might have (Getzel, 2008).

However, this is often not true.

There are basic accessibility tools on all computers and phones like the ability to change text size, visual effect, magnifiers, color filters, and screen readers. These tools are not made with neurodiversity in mind. Instead, they are often made for people with physical disabilities. The needs of those with physical disabilities are different from neurodiverse people even if the needs might be similar. Consider someone who is visually impaired. Most people who are visually impaired still have vision. Only about 15% of visually impaired people are completely blind (Lee & Mesfin, 2024). This person would most likely need to change the text size to be larger, use screen magnifiers, and a screen reader. Now consider someone who is dyslexic. Someone who is dyslexic might change the font to one that helps their reading comprehension and use text to speech. One who is not visually impaired or dyslexic might see the need of a screen reader and use of text to speech as the same problem. However, dyslexics do not need a screen reader to read everything on screen. They often just need to have the software read a specific block of text. Therefore, text to speech software like ChromeOS's select and speak can be particularly useful (Bone & Bouck, 2017).

Often the accessibility tools that are built into the systems are not helpful to neurodivergent students. Other assistive technology can also be inaccessible to college students due to the cost being generally very expensive (C. Smith & Hattingh, 2020). Therefore, the students have to rely on the technology provided to them through the school's disability services even if that technology is also not made with neurodivergent students in mind. Access to assistive technology is essential to neurodiverse students as it leads to more success in college and therefore can improve career outcomes (Getzel, 2008).

2.2.4 Teaching Neurodivergent Students

Research has been done to discuss how neurodivergent students use accommodations or modifications to help them learn. While accommodations are crucial for the success of students, they are often not enough (Newman et al., 2011). Grading criteria and course policies are often ableist (Birdwell & Bayley, 2022). Disabled students often fail classes because they cannot show the knowledge gained in the way that is expected. Grades and policies that are based on attendance, participation, presentations, and group work put neurodivergent students at a disadvantage.

It is important to understand that neurodivergent students learn differently. Students with learning disabilities benefit from the use of specialized support and technologies and flexible learning opportunities (Couzens et al., 2015; Hillier et al., 2018). Students with ADHD benefit from interactive teaching approaches, group work, and tutoring (Clouder et al., 2020; DuPaul et al., 2017). Autistic students benefit from individualized attention and practical activities (Barnhill, 2016; Van Hees et al., 2015). Dyslexic students can benefit from tactile learning (Cipolla, 2018). Neurodiverse students benefit from alternative but equivalent forms of assessments such as multiple choice and matching questions on exams (Clouder et al., 2020). Autistic students do better on assessments when they have access to extend time, distraction free testing spaces, and flexible due dates (Sarrett, 2016).

Sometimes being neurodivergent means that normal classroom assessments do not actually show how much knowledge a student actually has (Ketterlin-Geller & Johnstone, 2006). Therefore, instructors need to consider what they want students to actually know versus whether or not the student can show it with the assessment. Lastly, it is important to consider how accessible classroom material is (Cohen et al., 2005). If there is a dyslexic student that needs a text to speech, then physical documents might not be accessible. If

the material is only digital, there is still the possibility that a text to speech will not work with the format. However, it is also important to look at how easy it is for instructors to adapt the material. If it is too much work, instructors might not have the time or willingness to adapt their material to this universal design (Rao & Gartin, 2003).

Neurodivergent students tend to thrive when they are given work that integrates theory and practice (Clouder et al., 2020). Dyslexic students work well with nontraditional educational approaches like tactile learning (Cipolla, 2018). Dyslexic students benefit from support both outside and in class, exam accommodations, and assistive technology (Nightingale et al., 2019). Dyslexic students also benefit from seeing material early, extra time on coursework, and recorded lectures (Clouder et al., 2020). Only providing academic accommodations for autistic students is often not enough. Accommodations for sensory and social deficits are also helpful (Van Hees et al., 2015). Autistic students thrive with more one on one teaching approaches (Van Hees et al., 2015). Students with ADHD often benefit from interactive learning and group projects (Clouder et al., 2020).

2.2.5 Attitudes of Instructors about Neurodiverse College Students

An important factor in a student's ability to succeed is for faculty to understand what disabilities a student might have (Rao & Gartin, 2003). This understanding allows for students to receive the accommodations they need without having to explain and convince an instructor that they do need the accommodation. It also allows for students and faculty to come to solutions faster when a student is experiencing an issue. Generally, instructors are aware that they are legally required to provide accommodations for students with disabilities (Zhang et al., 2010). However, instructors can still unknowingly have harmful ideas and opinions on a student’s disability.

Most instructors have no problem with allowing testing accommodations for students. However they might also have the opinion that by providing this accommodation, they are not preparing students for the real world because employers might not be willing to give them extra time to complete a project (Rao & Gartin, 2003). This shows a fundamental misunderstanding of disability. The instructor is drawing a conclusion that the school environment is the same as a work environment, but this is simply not true. An exam is an assessment of the knowledge of a student. An exam accommodation is given so that the assessment of knowledge is not hindered by the student's disability. Real world tasks are more dimensional than exams. Real world work relies on multiple different people and can be delayed for a multitude of reasons. Having extra time on an exam cannot predict how well a student will do in a real world work environment.

Instructors may believe that they should be allowed to provide accommodations based on the specific disability and the severity (Rao & Gartin, 2003). However, instructors are rarely knowledgeable enough to make these decisions. This assessment has already been done by disability services, the student, and other professionals. It is not possible for an instructor to know what a student needs and does not need based on what they present themselves as to instructors. Students might be masking or downplaying the severity of their disability as to not have others react in a way that makes them uncomfortable. If a student is granted an accommodation, they need that accommodation. The instructor does not need to know why nor should they be the authority on whether or not the student actually needs it.

Comments about the student not needing accommodations can be extremely demoralizing. Consider a student that has used accommodations for most of their academic career. Since they were able to use these accommodations, they were able to do extremely well in classes. At the start of the semester, the student provides their accommodation

letter and the instructor approves their accommodations. For this particular class, they have a take home exam, so they don't need to take their exam in the university provided distraction free testing environment. However, they do still need their extra time. At the top of the exam, all students are supposed to write down how long it took them to complete the exam. They have two hours, so this student is granted four as they have double time accommodation. After class, the student went to remind the instructor that they do receive double time so as to not have any problems when they see that the student took longer than two hours on the exam. The instructor replies 'you don't need it but fine'. It's humiliating for a student to expect to just remind an instructor that they have an accommodation and be told they don't need it. To many neurodivergent students being disabled is a part of their identity (Shmulsky et al., 2021). If you remove their neurodivergence, you remove the essence of what makes them uniquely them. While often not the intent of the instructor, comments like these can be incredibly harmful.

There is a need for instructors to be exposed to knowledge about disabled students as instructors might not feel fully equipt to support and interact with disabled students (Zhang et al., 2010). Everyone has their own biases, so it's important for instructors to be exposed to knowledge about how disabilities affect students. Instructors will often feel more comfortable if they have sufficient support from the university's disabilities services, department administrators, and support staff (Zhang et al., 2010). However, university's disability services are often inadequate at educating faculty and staff about disabilities. In order for all instructors to be educated on disability, there would need to be mandatory training run by disability services. However, it is often hard to get instructors to actually agree to mandatory training due to scheduling (Zhang et al., 2010). Online training that is self-paced can be an option, but many faculty will speed through the training because they cannot fit training into their schedules.

2.3 UDL Framework

One way to better provide a supportive environment for neurodiverse students is to design courses with Universal Design in mind. Universal design refers to the process of designing products or infrastructure that is accessible to all people instead of adding accessibility features after the product or infrastructure is in place (Abascal, 2002). An example of universal design in the physical world is curb cuts (Schreffler et al., 2019). Curb cuts were originally created with accessibility for wheelchair users in mind. However, parents with strollers, bikers, shoppers with shopping carts, and more all benefit from curb cuts. Curb cuts also don't hurt the mobility of people that are just walking along the sidewalk or cars driving on the road.

The Universal Design for Learning (UDL) framework takes this idea and applies it to planning classes. UDL is supported by research from neuroscience, instructional design practices, and learning sciences (F. G. Smith, 2012). It is built off of three primary neural networks that lead to effective learning: recognition, strategic, and affective (F. G. Smith, 2012). The recognition pathway involves identifying, recognizing, and seeing patterns. The strategic network allows for setting goals, developing plans, and acting on those goals and plans. The affective network allows for an understanding of why the material is being taught and learned.

2.3.1 Universal Design

The idea of Universal Design was created in the 1970s by Ronald Mace at the Center for Universal Design at North Carolina State University (Scott et al., 2003). Universal Design is the idea of being intentional about designing and producing products that can be used by everyone (Abascal, 2002). The concept of universal design forces a designer to consider and avoid common accessibility barriers. It requires an awareness of human diversity,

anticipation of needs, and a drive to provide an inclusive environment (Scott et al., 2003). Universal design makes it so that there are fewer modifications that need to be made after a product is produced (Akoumianakis & Stephanidis, 2001). These designs then go on to help more individuals that just the group the adaptation was made for. Another important factor to consider when using universal design is to not create another accessibility issue when trying to accommodate for a different problem.

Universal design improves the usability of products (Abascal, 2002). Consider voice assistants like Google Assistant, Alexa, or Siri. For someone who is visually impaired using these assistants might be the only way the person can use their phones. However, these assistants help nondisabled people under special conditions like when they are driving or cooking. Universal Design cannot solve all accessibility problems nor can a designer predict all issues that could possibly arise (Abascal, 2002). However, using universal design to produce more accessible technology helps everyone navigate the world easier. Designing with universal design has many long term benefits besides allowing everyone to use a product. Accessibility modifications are often required by law. Creating a product and then trying to modify and maintain it to be accessible is costly (Stephanidis et al., 2001). For example, there can be technical issues that make updates harder to implement. Therefore, it is better to design around common barriers first to make accessibility features easier to implement later on.

2.3.2 Universal Design for Learning Framework

Universal design for Learning (UDL) uses the concept of universal design to create a better learning environment instead of a product. UDL is a framework that can be used to create a more inclusive classroom (F. G. Smith, 2012). The idea of the framework is to "design features of instruction that are essential to some students, beneficial to others, and not detrimental to any" (Hansen et al., 2016). Using UDL can address legitimate student

needs while protecting the integrity of the course and promoting learning (Scott et al., 2003).

UDL anticipates students' needs instead of reacting to them. Trying to accommodate students after a course has started is often time-consuming and difficult to implement (Scott et al., 2003). This allows for more students to be helped as the instructor is anticipating what students need. UDL does not lower the integrity of a course. Instead, it allows for more students to succeed by removing obstacles that would prevent the student from being able to learn effectively. Instructors can implement UDL by recording lectures, creating multiple formats of material, allowing students to access material early, providing outside resources, creating interactive learning opportunities, and adding optional group or individual work (Clouder et al., 2020).

Universal Design for Learning is a planning process, instead of tasks that must be done or rules to be followed (Israel et al., 2020). This planning process's purpose is to actively plan aspects of a class to be accessible and engaging to as many students as possible rather than creating modifications or accommodations to make it more accessible after policies are in place. Since there is no exact protocol for an instructor to follow, an instructor instead needs to focus on the objective of the course and the uniqueness of their students. These factors will then help the instructor to properly apply the UDL framework to the course.

There are multiple different ways to implement UDL principles. UDL is mostly studied in the context of K-12 classrooms. In this dissertation Scott et al.'s principles were used as they are principles made specifically for adult instruction. Scott et al proposed nine UDL principles that help to guide the planning process: equitable use, flexibility in use, simple and intuitive, perceptible information, tolerance for error, low physical effort, size and space for approach and use, a community of learners, and instructional climate (Scott et al.,

2003). Table 2.1 briefly gives a definition for each of these nine principles. These principles were created specifically for postsecondary education.

Table 2.1 Universal Design for Learning Principles

UDL Principle	**Definition**
Equitable Use	Instruction is designed to be useful and accessible to all students
Flexibility in Use	Accommodate all student abilities and provide them with multiple different methods of learning
Simple and Intuitive	To teach in a straightforward way that eliminates as much unnecessary complexity as possible
Perceptible Information	To communicate necessary information in a way that would be effective for all students
Tolerance for Error	Requires an instructor to anticipate variations in students' learning, pace, and skills
Low Physical Effort	To design to minimize nonessential physical effort in order to focus on learning the material
Size and Space for Approach and Use	Design an environment that is appropriate for all students no matter the student's needs
A Community of Learners	Requires an environment that promotes communication among students and between students and instructors
Instructional Climate	To design the course to be welcoming and inclusive

Equitable Use:

Equitable use means that instruction is designed to be useful and accessible to all students (Scott et al., 2003). An example of equitable use in a college programming course could be providing the sample code that was used in class. It is useful and accessible as it allows all students to go back to code seen in class to help them with programming projects.

Flexibility in Use:

Flexibility in use is meant to accommodate all student abilities and provide them with multiple different methods of learning (Scott et al., 2003). In a programming course teaching loops, this could look like having a lecture that introduces loops, a lab that uses a loop, homework that traces a loop on paper, and programming projects that use multiple loops.

Simple and Intuitive:

Simple and intuitive means to teach in a straightforward way that eliminates as much unnecessary complexity as possible (Scott et al., 2003). In a programming course, this could look like providing students with clear expectations of what their code needs to include without assuming a student's experience or knowledge. For example if an instructor asks students to write a programming project that simulates the Pokémon games. For those that have never played Pokémon before they would need to know how the game works. For those that have played Pokémon, they would need to know if they need to add items, move probabilities, status conditions, ect.

Perceptible Information:

Perceptible information means to communicate necessary information in a way that would be effective for all students (Scott et al., 2003). In any course, this could look like having the date for exams in multiple locations including reminding students in class, adding dates to the syllabus, and making a note on the learning management system used for the course.

Tolerance for Error:

Tolerance for error requires an instructor to anticipate variations in students'

learning, pace, and skills (Scott et al., 2003). In a programming course, this could look like allowing students to submit code before an assignment is due inorder to receive feedback.

Low Physical Effort:

Low physical effort means to design to minimize nonessential physical effort in order to focus on learning the material (Scott et al., 2003). In a programming course, this can be applied by allowing students to use an IDE when writing code. This would allow students to focus on learning the concepts of programming rather than particulars like if the syntax is Try{} Catch{} or try{} catch{}.

Size and Space for Approach and Use:

Size and space for approach and use means to design an environment that is appropriate for all students no matter the student's needs (Scott et al., 2003). An example of this is changing the seating position to accommodate students. For college courses this is often out of the instructors control. An instructor can request a specific room like one with circular tables instead of row desks or a time slot, but there is simply no guarantee that the request would be fulfilled.

A Community of Learners:

A community of learners requires an environment that promotes communication among students and between students and instructors (Scott et al., 2003). In any course, this could look like encouraging students to work together during an in class assignment.

Instructional Climate:

Instructional Climate means to design the course to be welcoming and inclusive. In an introductory programming course, this could mean highlighting diverse individuals that were important to the field of CS. For example, the first computer programmer being Ada

Lovelace.

2.3.3 Using UDL in College Courses

Often UDL is discussed and studied in the context of a K-12 classroom. However, UDL can be used in any educational setting with the modification of different considerations being taken into account (Scott et al., 2003). In K-12 classrooms, there are often mandated curriculum or content standards. In college, courses differ from school to school and from instructor to instructor. As a result, there is greater flexibility for an individual instructor to apply UDL principles in a college course. However in order for this to be done properly, instructors need to be educated on UDL. This is often difficult as there is not a mandatory requirement for instructors to learn different instructional strategies (Scott et al., 2003).

Universal Design for Learning is not often used or studied in STEM courses in higher education (Schreffler et al., 2019). However, it is needed as there has been an increase in neurodivergent students enrolling in STEM majors (Schreffler et al., 2019). Using UDL in college courses allows students to personalize their learning which helps to provide more opportunities to engage with the course (F. G. Smith, 2012). This helps to create an environment that motivates students and increases their interest in their own education (Schreffler et al., 2019).

2.3.4 UDL Efficacy for Disabled and Neurodivergent Students

Often a modification using UDL will help students that would possibly not have been diagnosed and neurotypical students. For example, using fonts that have been shown to help dyslexic students are also beneficial to nondyslexics (Rello & Baeza-Yates, 2016). By using UDL, a dyslexic student's disability is minimized as elements like multiple formats of instructions, optional group projects, peer mentorship, digital materials and varied teaching approaches help to mitigate the symptoms of dyslexia (Clouder et al., 2020).

UDL can also help remove negative emotional responses that neurodivergent students have (Clouder et al., 2020). UDL removes the need to rely on outside accommodations and by consequence removes the need for students to disclose their disability if they are uncomfortable (Clouder et al., 2020). If an instructor keeps in mind universal design, students will need accommodations less and less as those accommodations are built into the design of the course (Ketterlin-Geller & Johnstone, 2006). However, universal design can never be perfect for all students, so students will always need accommodations to get the most out of a lesson (Hansen et al., 2016). It's important when designing course material to assess possible accommodations students might need and how to implement them if those needs arise.

When using the UDL framework it is also important to understand that neurodiversity has a culture. Like any culture, neurodiversity culture has preferred communication and expression (Gobbo & Shmulsky, 2020). When many neurodivergent students get to college, their neurodivergence is an important part of their identity, and by instructors invalidating that identity students will have decreasing levels of self-esteem, less ability to cope, and worse mental health (Shmulsky et al., 2021). By taking this into account, courses can be designed to not force students out of the comfort of their culture norms which will allow them to feel a greater sense of belonging in the classroom and a safer environment.

2.3.5 Instructors Implementing UDL

Instructors that are already seen as good teachers often already have implemented UDL principles based off of past trial and error (Scott et al., 2003). For some instructors, Universal Design for Learning will require instructors to think about courses differently. Often beginner courses like introductory programming courses are seen as "weed out" courses that are meant to not allow "unfit" students from continuing pursuing a STEM

degree. UDL forces instructors to reject the idea that there are students that do not belong in a major (Scott et al., 2003).

Some instructors believe that designing courses with UDL can be valuable as it can help more students (Scott et al., 2003). However, some instructors are more skeptical of UDL as there is no single best way of teaching for all students (Scott et al., 2003). This is the wrong way to think about UDL. The UDL framework allows for multiple ways for students to learn. If one way is not the best way for a particular student, there are other built-in ways to learn the same material differently. In order for UDL to be done properly, instructors need to be educated on what it is and how the framework is applied. Understanding UDL can give more structure to instructors, so they can feel more confident about making changes to their courses (Scott et al., 2003).

2.4 Teaching Programming

Computer science skills have increased in necessity. Computer science and computer science education changes quickly which has led to many issues arising when creating a curriculum. Curriculums will often differ in the choice of language and content (Mehmood et al., 2020). While introductory programming courses are seen as essential, there is often a disagreement about which language should be taught (Mehmood et al., 2020). Thus, what is taught in an introductory programming course can differ across different universities (Mehmood et al., 2020). However, generally, an introductory programming course refers to a course that focuses on problem solving skills, basic programming concepts, syntax of specific programming language and creating simple programs to solve simple problems (Medeiros et al., 2019). Even though most students struggle with introductory programming concepts, neurodivergent students are at a particular disadvantage. Therefore, appropriate changes should be made to instruction to help neurodivergent students. As a consequence,

neurotypical students also benefit from improvements made to teach programmers with disabilities (Haynes-Magyar, 2024).

2.4.1 Why Teaching and Learning Programming Is Difficult

Programming has become an important skill in any field but especially in STEM fields. There has been a lot of research that has been done in order to assess what the best way of teaching and learning programming is (Robins et al., 2003). However, there is still a huge dropout and failure rate when it comes to introductory programming courses (Medeiros et al., 2019). It's not clear why this is happening, but possible theories include programming being a difficult skill to learn, the complexity of the content, teaching pace, the instructor having poor teaching skills, engagement with students, or poor programming skills (Wells et al., 2012).

Students need to have both programming related and general education skills to succeed in introductory programming courses (Medeiros et al., 2019). The programming related skills a student often must have include: problem solving, mathematical ability, and abstraction (Medeiros et al., 2019). Students need problem solving skills so they can dissect the important information needed to create a program that solves the particular problem that is given to them. Mathematical ability is important as a lot of programming relies on the use of logic and arithmetic. Abstraction refers to the ability to define patterns, generalize specific instances, and create parameters (Medeiros et al., 2019). This is often a necessity as creating functions or methods require generalization so the function or method can be run in multiple instances of a problem.

The general educational skills a student must have include: critical thinking, discussion skills, creativity, and time management (Medeiros et al., 2019). Critical thinking is important as programming requires students to be able to interpret and analyze different

problems and create solutions using code. Discussion skills are important as often programming is not an independent task. Multiple people are often working on a single project. Thus, it is important for a student to be able to discuss their work in a succinct and technical manner (*Learning to Write and Writing to Learn: Integrating Communication Skills into the Computing Curriculum | IEEE Conference Publication | IEEE Xplore*, n.d.). Creativity is important as there is no one correct way to program. Therefore, students often need to come up with complex but creative solutions to a problem (Ambrósio et al., 2011). Time management is important as programming projects are often not as simple as doing separate homework problems. Therefore, students need to be able to efficiently plan, write, and debug their solution in the limited time that is often given (Beaubouef & Mason, 2005).

The difficulties of teaching programming generally fall into three categories: the curriculum, teaching and learning, and assessments (Mehmood et al., 2020). When discussing the curriculum, the language choice and content are often not agreed upon. When choosing a language, instructors often believe that teaching a syntactically simple language that is also useful in industry will help students focus on learning programming concepts (Mehmood et al., 2020). Therefore, often Python or Java is used as a beginner programmers first language.

Often the curriculum is designed specifically for a particular computer science student's degree requirements, even though many STEM majors now require introductory programming as part of their degree requirements (Mehmood et al., 2020). This lack of standardization has often caused issues when students graduate as industry cannot successfully predict a recent graduate’s knowledge. Another issue when creating computer science curricula is that computer science is a fast paced field meaning that often languages and technology become obsolete quickly. This makes it difficult for instructors to properly predict what a student needs to know before they graduate.

Research has investigated what the best methods and techniques are when it comes to both teaching and learning programming. Most research tends to focus on creating teaching methods and tools rather than student challenges like programming related and general education skills (Medeiros et al., 2019). There is less of an emphasis on analyzing and evaluating the tools and methods that are created (Mehmood et al., 2020).

There is no consensus on what the best way of measuring students' knowledge in introductory programming courses is (Mehmood et al., 2020). Often the rubric for assessing students is based on the functional correctness of the written code, the technical correctness of the written code, and current code conventions (Mehmood et al., 2020). There is often a discussion as to what the best method of assessment is. These methods can include coding on paper, coding in an IDE with a compiler that students can use to debug during the assessment, and coding on the computer but without the use of a compiler (Kurniawan et al., 2020).

2.4.2 Teaching Programming with Animations and Images

It is common for pictures and other visualizations to be used to help reinforce the material being taught (Bryne et al., 1999). Visual learning can help students think of new ways to solve problems, provide alternative ways of thinking and reinforce topics and practice in science and engineering classes (McGrath & Brown, 2005).

In many computer science classes instructors will often use diagrams and other images to enforce learning. This can help beginning programming students understand how written code turns into a dynamic process (Sorva et al., 2013). However, static visualizations can only give a basic understanding of how something works. There have been many different code visualization tools that have been created for various languages

(Sorva et al., 2013). However, many of these tools are only prototypes and, therefore, are not practical to use in the classroom (Sorva et al., 2013).

Instructor made animations, meanwhile, can help explain how a process works dynamically (Bryne et al., 1999). Creating small animations would allow an instructor to control how the code is being visualized to better reflect their teaching. Using small animations can often be more practical as becoming obsolete will not be as difficult to update unlike code visualization tools (Sorva et al., 2013). There are two types of computer animations: user-controlled and free running. User-controlled animations allow students to start and stop the animations. Sometimes this also allows students to change the speed of the animation. Free running animations do not allow students to control the animation. Both of these types of animations can help students understand both complex and abstract topics (Kuyath, 2002).

The use of animations as an aid to help students understand computer science topics, like algorithms. Two types of tools have been studied in order to incorporate animations in computer science education: Visualization systems and animations made in order to demonstrate a concept (Végh & Stoffová, 2017). Visualization systems can help students with understanding and modifying their own code. However, they have the disadvantage of being general which leads every program to be displayed in the same way. This can lead to a lack of understanding of the critical characteristics of specific algorithms. The drawback of using animations made specifically for a concept is the amount of time it takes to create. However, using these types of animations allows for a better understanding of important characteristics of an algorithm, can better focus on specific steps, and contain more interactive elements (Végh & Stoffová, 2017). In general, students like using animations to learn, but they are not effective in all cases (Végh & Stoffová, 2017). In order for animations to be effective, animations need to be graphically well-designed and need to

be interactive so that students can actively participate in the material (Végh & Stoffová, 2017).

2.4.3 Teaching Programming using UDL

Teaching Programming using Universal Design for Learning has not been studied in university level computer science courses. Most work has been done in the K-12 space. However, the general principles of UDL in CS courses can be applied to any computer science course and often looks similar to implementing UDL in any other course context (Israel et al., 2020). Often instructors interpret and apply UDL too narrowly and thus need a lot of support in order to implement UDL in CS courses (Israel et al., 2020). Applying UDL to CS courses needs to ensure that there are multiple means of engaging and motivating students, representing content, and action and expression (Israel et al., 2018).

There has not been much research in the area of adjusting a computer science curriculum to help the diverse students that vary in skill level (Hansen et al., 2016). There are many reasons why students in programming courses will have varying skill levels including previous exposure to computer science, variations in K-12 experiences and opportunities, disability, English proficiency, and going into a non CS related major that requires programming courses (Hansen et al., 2016). In the K-12 classroom, students would often ignore written instructions, hints, and feedback (Hansen et al., 2016). At the college level, often programming projects are written with multiple pages of requirements and explanations. A way of helping students that struggle with text is to eliminate large complex explanations and instead simplify instructions by underlining and bolding key words and embedding text into interfaces (Hansen et al., 2016). This helps students to interpret what information is important, so they can spend more time working on programming.

A common skill that varies in an introductory programming course is varying math skills. Since math is important to computer science, a UDL change can be to first remind or teach a student particular math concepts that they might not have seen before and then explain how it should be implemented in code (Hansen et al., 2016). This helps students with weaker math backgrounds to not be overwhelmed with learning two concepts at the same time.

A common mistake that computer science instructors will make is the 'everyone knows' fallacy (Hansen et al., 2016). Computer science allows instructors to be creative with their projects that often leads to students being more engaged with the material. Unfortunately, this often leads to instructors not realizing that students might not have been exposed to something that the instructor is trying to have them implement (Hansen et al., 2016). Neurodiversity in recent years has developed into a culture (Gobbo & Shmulsky, 2020). Similar to other cultures, neurodiverse students will have different experiences than neurotypical students. Therefore, it is important for an instructor to either try and create projects that are universal to their students or make sure anything that they are requiring to be implemented can be picked up quickly.

2.4.4 Teaching Programming to Neurodivergent Students

When discussing the need to make CS more inclusive, research tends to focus on racial and gender minorities. However, there is a need for more disabled programmers as well. Neurodiverse students often do not go into computer science for many reasons including inflexible and accessible learning materials (Haynes-Magyar, 2024; Ladner & Israel, 2016). Often research discussing how to teach neurodiverse students programming, looks at teaching programming using drag and drop block coding. The reason for this being that block coding improves problem-solving abilities, lowers the cognitive load needed to code, and teaches coding patterns (Haynes-Magyar, 2024). However, block coding is not

what computer science students are going to be taught in college courses. Therefore, other methods of teaching programming need to be studied in order to consider what the best methods of teaching programming to neurodivergent students are.

Not a lot of research has been done on how different methods of teaching programming can help neurodivergent students. Most of the research has also focused on K-12 neurodivergent students instead of college students. The most well researched technique when it comes to neurodiverse students is visual programming. Visual programming has been shown to not be accessible for children with learning disabilities and cognitive impairments (Zubair et al., 2021). Current visual programming languages are also not very accessible for children with ADHD (Galeos et al., 2020). However, dyslexic adults were found to be more productive if they were using a visual programming language (Fuertes et al., 2018). Visual programming has also been shown to improve computational thinking skills in autistic children (Munoz et al., 2018). Therefore, visual programming might be beneficial to some neurodivergent students, but could be inaccessible to others (Clouder et al., 2020). One study has shown that game based learning does help neurodiverse students with computational thinking (Asbell-Clarke et al., 2021). The studies on visual programming and game based learning might suggest that the use of animations and images while learning could be beneficial to neurodivergent students.

While the use of animations and images in the context of helping neurodivergent students learn programming has not been studied, the use of animations to help students with disabilities has become increasingly popular to help with various basic skills including reading, writing and math skills (Baglama et al., 2018). However, often these studies focus on "fixing" the symptoms of the neurodivergent condition rather than aiding in learning. For example, animations have been used to help autistic children communicate (Mulholland et al., 2008). Animations were found to be more useful than static materials for both college

students with dyslexia and non dyslexic students (Taylor et al., 2007). Hypermedia, which includes both animations and pictures, has been used to help students with ADHD retain knowledge (Fabio & Antonietti, 2014). A study has also shown that students with ADHD performed significantly better on animated computerized tasks (Shaw & Lewis, 2005).

Neurodivergent students often experience accessibility barriers when it comes to their computer science courses. Not all parts of computer science courses are conducted on the computer. Therefore, assistive technology that neurodiverse students are generally used to using for assignments and class are often removed when it comes to exams and quizzes. A common exam and quiz format, especially for large institutions, for introductory programming courses is to have students code on paper (Kurniawan et al., 2020). Students often dislike this practice as they feel that they can code faster on computers and find coding on paper to be a useless skill that is only for an academic space (Kurniawan et al., 2020). For a dyslexic student, this practice is detrimental (Borsotti et al., 2024). The person evaluating a question on the exam often has no knowledge that the person that they are grading is dyslexic or if a mistake was made because a student is dyslexic or because they do not know the syntax of the programming language. Therefore, allowing students to use computers with assistive technology that all students are used to like spell check and compilers can truly tell what a student knows while also reflecting real world coding conditions (Borsotti et al., 2024).

Neurodivergent students also often experience that their assistive technology is not made for math and coding. Often screen readers are not configured to read math formulas and text written in IDEs (Borsotti et al., 2024). Assistive technology that has been designed to read math formulas or text written in IDEs are often not downloaded on computers that students with disabilities use in alternate testing centers. Thus, often neurodiverse students

have to struggle through at least one exam or quiz until the proper software can be downloaded for them to use (Borsotti et al., 2024).

Students with problems with executive function, like students with ADHD, often cannot get accommodations for these issues. Computer science classes often have fast paces that lack multiple forms of engagement and workload, so neurodivergent students often have difficulties paying attention to the entire class (Borsotti et al., 2024). Therefore, it is often a necessity for students to be allowed to record the class in some sort of way. This is often something that instructors are uncomfortable or unwilling to do. In courses, where instructors do record, instructors will often switch between slides and code in IDEs which makes the recordings difficult to follow. This often forces neurodivergent students to have to spend more time than their neurotypical peers to review all material learned in the course.

Chapter 3: Methodology & Methods

This chapter starts by reiterating the research questions. The study design is then discussed for each semester Fall 2022, Spring 2023, and Fall 2023. Next, there is an explanation of where the study took place, who the instructor of the course was, who the students were, and an explanation of each of the courses studied. Then, there is an explanation of how participants were recruited and the specifics about the participants. Lastly, the data and how the data was analyzed is explained.

3.1 Restate Research Questions

The study pursued the following questions:

- How can universal design for learning principles be implemented into an introductory programming course's materials?
- How does updating an introductory programming course to follow the universal design for learning framework affect the experience of students?
- What are the experiences of neurodivergent and neurotypical students after updating an introductory programming course to follow the universal design for learning framework?

3.2 Study Design

This study was conducted with the same instructor in two different introductory programming courses: Object-Oriented Programming I (CMSC 131) and Object-Oriented Programming II (CMSC 132) (Appendix A, B, C). CMSC 131 and 132 are the first two programming courses taught as part of the undergraduate computer science program and required for all students to take as part of the major. The study took place over three semesters Fall 2022, Spring 2023, and Fall 2023. In Fall 2022, CMSC 131 was used as the control. No changes to the course were made from how the course is historically taught. The

only modification for the control portion of the study was that students were asked to complete several surveys over the course of the semester. The data that was collected included a demographics survey, RAADS-R questionnaire, 4 after-exam reflection surveys, and grades.

In Spring 2023, lecture slides for CMSC 132 were updated to be more accessible with the guidance from Universal Design for Learning principles. The students were asked to complete two surveys: a demographics survey and the RAADS-R questionnaire. The students were then invited to participate in an interview which discussed their experiences in both CMSC 131 and CMSC 132. The instructor and teaching assistants for CMSC 132 were also invited to participate in an interview that discussed their experiences as teaching staff for introductory programming courses.

In Fall 2023, lecture slides for CMSC 131 were updated to be more accessible and supplementary slides were added to the programming projects with the guidance from Universal Design for Learning principles. The same data collection protocol was completed for the experimental portion of the study, meaning the data collected included a demographics survey, RAADS-R questionnaire, 4 after-exam reflection survey, and grades. At the end of the semester, students were invited to participate in an interview to discuss their experience in the course. The teaching assistants for CMSC 131 were also invited to participate in an interview that discussed their experiences as teaching staff for introductory programming courses.

3.3 Setting

This study was conducted at the University of Maryland (UMD) in College Park, Maryland. UMD is a public research university that hosts around 40,000 students total which includes both graduate and undergraduate students. Of UMD's population, 16,000, ~4%,

undergraduate and 1,500, ~2%, graduate students are registered with UMD's disability services office. The computer science department at UMD has one of the largest computer science programs in the US with about 3,200 undergraduates. UMD's computer science (CS) program is a limited enrollment program. This means that the major is very competitive and has special requirements. The study was conducted in two of UMD's introductory programming courses: Object-Oriented Programming I (CMSC 131) and Object-Oriented Programming II (CMSC 132). During the semesters the study was conducted in, all courses studied were taught by the same instructor. The participants for this study included the students, the teaching assistants for the courses being studied, and the instructor of the courses being studied.

3.3.1 UMD's Computer Science Department

UMD's computer science (CS) program is particularly competitive and stressful due to the computer science major having a limited enrollment program. Only about 600 first year students get admitted directly as computer science majors. First year students can be admitted to UMD and the CS major, admitted to UMD but not the CS major, or not admitted to the university. Those that are admitted to UMD but not the CS major are assigned the Letters and Sciences major, which is UMD's undeclared or undecided track. However, students may still be able to become CS majors. Around 100 transfer students are taken each year. These students include both internal and external transfers.

Students that are at UMD and still want to pursue CS can do so if they complete gateway requirements and another review process. The computer science gateway courses are the two courses used in this study: CMSC 131 and CMSC 132. During the semesters of this study, students had to receive a minimum grade of a C- in both beginning programming courses. Additionally, the students would also have to acquire a minimum grade of a C- in Calculus I and a minimum grade point average of 2.7 in all of the courses that were taken

at UMD or other institutions. They must then pass benchmark requirements which include three more specific courses with at least a C-, and a minimum grade point average of 2.0 in all courses taken at UMD and any other institutions.

All students in a limited enrollment program like CS need to follow guidelines or they are dismissed from the major. Only one gateway or performance review course can be repeated to earn the grade needed. Even if a student withdraws from a course it is still considered a repeat. Students are only allowed to apply once to a limited enrollment program. Any student that was directly admitted to the program and does not meet the performance review criteria are dismissed from the major and they cannot reapply. All students must also maintain a minimum cumulative grade point average of 2.0 or they will be dismissed from the major. This creates a competitive and stressful environment for students. Therefore, this study is important in order to help students continue the major they want to pursue.

3.3.2 Instructor for Courses Studied

During the semesters the study was conducted in, all courses were taught by the same instructor. The instructor for these courses is a principal lecturer in the Department of Computer Science at University of Maryland (UMD). At the start of the study, he had been teaching introductory programming courses for 25 years.

Before the Spring 2023, the instructor was interviewed about his teaching experience, introductory programming students, and neurodiverse students. The instructor believes that students struggle with programming as they come in with preconceptions of programming. So, students are often surprised when their original assumptions are incorrect. He stated “when they see the reality of it, they realize, this is not for them.” He tries to make sure that students are not dissuaded from the major because of a bad

experience but instead because they find out programming is not for them. He does believe that "some of them [students], I hate to say, they're not cut out to do it.". However, he would rather have students see programming as something that is not for them rather than something they cannot do.

The instructor also believes that students that struggle with introductory programming courses are struggling because of discipline and not any academic skills that they may or may not have gained in high school. He stated that "I don't think that you need to know math for [CMSC] 131 or even [CMSC] 132. You don't need calculus or even precalculus. I think it's more about discipline." The instructor finds that many students have bad habits that they have from high school that the students take with them during their first years at college. The instructor stated, "They need to adapt their habits because if they come from a high school, that was not as strict, they're gonna suffer because they have to change how they go about things in particular, the one main problem is that they wait until the last minute to submit things because that worked for them in high school and they were able to get A's but now you cannot do that here." He doesn't believe that introductory programming courses are more challenging than any other course.

The instructor's main complaint about introductory programming courses is the number of students taking them. Around 1,000 students take CMSC131 and CMSC132 each semester. He would like to give more individual help to students, but with the size of these courses at UMD it makes individualized help difficult. He often finds that the teaching assistants are better at being able to provide individualized help to students. He believes "those students that are that are having trouble picking up the material benefit from lab, those that have previous experience, maybe don't even attend lab which is normal."

The instructor believes that if students are never struggling with the material then they are not in the correct class. He stated "If I see a student that does not struggle at all, then the student should not be there in the first place, but some students try to play safe." He believes that students who are not struggling with material are not learning. He stated, "But if the student does not struggle, that means they're not learning. If they're not learning, that means that they already know it."

The instructor is open to changes led by the students taking his courses for later semesters when he teaches the same course. However, he doesn't ask for feedback from students during the semester so he can avoid major changes. He does however try and listen to students when it comes to small changes in the course. He stated "I'm always trying to listen to students so I always tell them 'email me if you have suggestions'. Usually I have a very informal mid-semester survey."

He also has a one question survey separate from the mid-semester survey that he gives out to students that asks whether or not they have had programming experience before this class. If the instructor sees that when distributing grades that students without previous programming skills are not doing as well as expected, he will take that into account when adjusting the course. An example of a change the instructor made to help students was to increase the number of exams. Before, the students had 2 exams and a final. Now students had 3 exams and a final. He did this so that "the student has more opportunities to do well throughout the semester. And even if they don't do well in the final, they can do well in the class."

In order to help students who struggle with discipline issues he changed the course so that many of the programming projects have two deadlines. He chose to do this as it will help them to get started. He stated, "that pushes them to start which sometimes is the hardest

part for some of them, some of them say ‘I don't know how to start’ or ‘once I start, I get in the rhythm of it’”. He also tries to finish the assignments that students are required to do to help students who “seem to cram a lot in the last two weeks of the semester when everything is due … so they can move on and begin to prepare for the final or work on other classes.”

The instructor doesn’t believe that students need to come to class to be successful. He believes “I think students, if the student watches the lecture, I record the labs. If they need help they can go to office hours or use Piazza[1]. They can be successful.” He believes that the students that come to class recognize that it creates some sort of discipline. If they did not come to class “then the videos begin to accumulate and have too many of them to watch”. He also believes that recording lectures is also helpful as “It's a great tool for the students that have problems with the [English] language.”

He tries to ease students into introductory programming courses by allowing the students to discuss the first few programming projects with each other. Since UMD’s introductory programming courses have a large number of students, students can struggle with talking to other students. The instructor does encourage the students on the first day of class to introduce themselves to each other, but often there are still students that are shy and would rather not talk to others. He believes that discussions can facilitate peer relationships easier. He stated “I think the lab discussions that we have can promote a little bit of that because it's a smaller setting, but the teacher can also promote that.” However, “I don't have something that in particular pushes them to work together.” In lecture, he does give breaks so students can talk to each other about material they are learning. He believes this time promotes peer interaction and helps students pay attention. He stated, “I do this often

[1] Piazza is an online, forum system that allows students to ask questions and instructors to post announcements

throughout the lecture; give more chances to people. Also for people that have problems with attention, it can help them because they don't have to be focusing all the time. It's a break."

The instructor does not curve his grades. He believes "that you [students] feel like if you put in the work, you get an A. That's very important in this course". The instructor believes that curves cause unwanted stress on students. He stated "You don't want to have a student that all semester has a 56, get an A at the end but the student has experienced all the stress throughout. I don't think that's helpful … Surprising them at the end by passing doesn't make sense." He believes that the students' mental state can greatly impact both their grades and the experience they have taking the course. He also believes that curving grades discourages students from helping others because it's going to hurt their grades by changing the curve.

In these classes, he has had accessibility and disability services students that needed accommodations. Generally, the students either gave the instructor their letter in person (pre-covid 19 pandemic procedure) or sent the letter to him electronically (post-covid 19 pandemic procedure) to be signed. The instructor does try to skim through the students' letters, but generally just lets the students inform him of any accommodation requests. He stated, "I try to skim through the letter. Okay. To see there's something in particular, but I let the student inform me of anything." The students in his introductory programming courses mostly request extra time on exams and quizzes, however he has also had requests for lectures captioned and for a particular student to always be able to sit in the front.

The instructor does not accommodate students beyond accommodation letters for fear of legal issues. The instructor stated "Many students approach me about this … I have not completed the ADS [accessibility and disability services at UMD], but because I cannot evaluate how much extra time they need I can get into legal issues." The instructor will

accommodate students with extensions if they have a short period of illness. He does include anxiety and depression episodes in his definition of illness. However, the instructor will not give extensions on a constant basis if the student has not gone to disability services to provide an accommodation letter.

In most cases, the students in his courses do not disclose their disabilities to the instructor. However, he had students disclose that they are struggling with anxiety and depression. The instructor stated that "I think this is something that happens a lot in our classes. I guess it has to do with the demands of the major. I think many students have this problem and it has never surfaced before because the classes had not been challenging. So they were able to cope with it." The instructor gives advice to students struggling with their mental health to reach out to their doctor and disability services to see if they can get help from professionals. He tries to help students by "emphasiz[ing] to them that they're not alone or that other students experience the same thing." He tries to give students hope while also accommodating them with an extension if needed. When students come to him, he tries his best to give helpful advice or provide resources to students as "it's a reality just because I cannot understand that I cannot dismiss it." The instructor believes "psychology is very important in this course. They are so demanding. Also, how competitive it could be. That's one of the problems they are always facing. You have a couple of students that are not supposed to be there. They already know the material that intimidates the people. It erodes the confidence of other people."

3.3.3 Students

University of Maryland's first two introductory programming courses are Object-Oriented Programming I (CMSC 131) and Object-Oriented Programming II (CMSC 132). A majority of the students taking these courses are in their first year of undergraduate

studies. However, the course often has both second and third year undergraduate students. Mostly Computer Science (CS) majors and Letters and Sciences majors take these courses. However, other majors including information science and mathematics majors may also take the course. Since the CS major is competitive many students come from backgrounds that include taking multiple math and computing related courses in high school. Many students have previously been exposed to programming languages including Java, Python, and HTML. These students often also took Advanced Placement (AP) level and dual enrollment classes. The students were often involved in computing related clubs and activities. They are often expected to have many extracurriculars, research, honors, and awards while in high school.

During the Fall 2022 semester when the study took place, about 22% of CS undergraduates were female with the remaining 78% of the students being male. The majority of the students were Asian and White. About 45% of undergraduate CS students were Asian and around 27% of students were White. The rest of the undergraduate student body consisted of about 8% being Black or African American, about 5% being Hispanic or Latino, about 4% being of two or more races, 6% were non US residents, and 7% were of unknown race.

3.3.4 CMSC 131: Object-Oriented Programming I

Object-Oriented Programming I (CMSC 131) is an introduction to programming course. It is often the first programming class that a CS student at University of Maryland takes. This course has a focus on object-oriented programming languages, specifically Java. The goal of the course is to develop skills in program design and implementation of those programs using an integrated development environment (IDE). The IDE used for this course was Eclipse. The students taking this course would either have already taken Calculus II or be taking it in the same semester as CMSC 131.

This course runs for 16 weeks. It meets three times a week for 50 mins for lectures. The course also has discussion sections that are run twice a week for 50 mins. The lecture is led by the instructor while the discussion section is run by teaching teaching assistants. There are two types of teaching assistants: grading TAs and teaching TAs. Both types of TAs conduct office hours.

The lectures are in large lecture halls that hold about 200 - 300 students for each lecture. The discussions are much smaller. Each discussion section has about 35 students in them. However, the students are not required to go to discussion, so often the discussions themselves have a lot less students in them. The instructor and TAs record all lectures and discussions.

The topics that are covered in CMSC 131 (Appendix A and C) are an introduction to computer systems, variables and operators, expressions, statements and methods, java text input and output, conditionals, loops, principles of object oriented programming, basics of program design, testing and debugging, java memory maps, arrays and java arraylists, java interfaces, inheritance, and recursion. There is no textbook to go along with the course, however, the instructor does link to some online textbook references for students to use as needed.

The grades for the course are based on programming projects, exercises, and lab work, quizzes, three exams, and a final exam. The programming projects, exercises, and lab work are worth 39% of the grade. Quizzes are worth only 3% of the grade. The three exams are worth 10% for Exam 1, 16% for Exam 2, 16% for Exam 3 for a total of 42% of the final grade. The final exam then makes up the last 16% of the total grade (Appendix A and C).

All assignments are due at 11:55 pm the day that the assignment is due. However, students are allowed to submit work a day late with a 12% penalty. After that, students are not allowed to submit for credit. All assignments are submitted through a project submission server specifically designed for UMD CS courses. Programming projects have release, public, and secret tests attached to them. These tests are the main part of the grade for the projects. The second part of their grade is to manually grade for good programming style.

The course uses Piazza for class communications. Piazza is an online, forum system designed for CS courses that allows students to ask questions and instructors to post announcements. A student can ask a question and other students, TAs, and instructors are able to answer the student. From there, a follow-up question or discussion is able to be asked. There are also different options for students to ask questions. Students are able to ask questions anonymously if they do not feel comfortable asking with their name attached to the question. Students are also able to post private questions that only go to specific people like the instructor and teaching assistants. There is also a rating system that allows students, teaching assistants, and the instructor to vote for good questions and good answers, which allows more students to see the question if it is important or for the students to see that another student's answer is fully correct and endorsed by the teaching staff.

The course uses Piazza for class communications. For CMSC 131, there are course-specific rules defined by the instructor. For example, students are not allowed to post code on Piazza. If a student does post code, they may receive an XF in the course. An XF in the course is a failure due to academic misconduct. The students are also not allowed to request or provide information or suggestions on how to implement anything for the project. They

are not allowed to request or provide information on how to pass the public or release tests that are attached to the programming projects.

The students are also responsible for checking class announcements on Piazza. They are allowed to ask questions about Eclipse, how to submit a project, or where the description of the project, examples or any resources can be found. They are also allowed to ask clarifying questions about concepts or examples that were done in class and administrative issues like class cancellations, deadlines, and office hours.

3.3.5 CMSC 132: Object-Oriented Programming II

Object-Oriented Programming II (CMSC 132) is the second introductory programming course at University of Maryland (Appendix B). If a student has previous programming knowledge, they might take the exception exam for the course CMSC 131, so CMSC 132 might be the first programming course that the student takes. CMSC 132 focuses on design, building, testing, and debugging medium-sized software systems. The students learn about object oriented methodology, algorithms, and data structures. CMSC 132, like CMSC 131, uses Java as the programming language of instruction. The students taking this course must have earned a minimum grade of a C- in CMSC 131, earned a 5 on the AP computer science A exam, or earned a 70 or above on the first and second part of the CMSC 131/132 exception exam.

CMSC 132 runs for 16 weeks. It meets three times a week for 50 mins for lectures. The course also has discussion sections that are run twice a week for 50 mins. The lecture is led by the instructor while the discussion section is run by teaching teaching assistants. There are two types of teaching assistants: grading TAs and teaching TAs. Both types of TAs conduct office hours.

The lectures are in large lecture halls that hold about 200 - 300 students for each lecture. The discussions have much less students in them. Each discussion section has about 35 students in them. However, the students are not required to go to discussion, so often the discussions themselves have a lot less students in them. The instructor and TAs do record all lectures and discussions.

The topics that are covered in CMSC 132 are abstraction and encapsulation, enumerated types, comparable, inheritance, recursive algorithms, debugger, memory maps, abstract classes, exceptions, testing, correctness, cloning, constructor/destructor, initializing blocks, levels of copying, intro to OO design, nested types, lambda expressions, event-driven programming, JavaFX, generic programming, collections, linear data structures, restricted abstractions, algorithmic complexity, hashing, sets and maps, linked-lists, file input and output, trees, heaps, priority queues, threads, synchronization, graphs and graph traversals, design patterns, dijkstra's algorithm, graph implementation, sorting, algorithm strategies, and software development. There is no textbook to go along with the course, however, the instructor recommends the textbook *Data Structures & Abstractions with Java, 5th Edition* by Frank Carrano and Timothy Henry. The instructor also links to more online textbooks for students to use as needed.

The grades for the course are based on programming projects, exercises, lab work, quizzes, three exams, and a final exam. The programming projects, exercises, and lab work are worth 38% of the grade. Quizzes are worth only 4% of the grade. The three exams are worth 14% for Exam 1, 16% for Exam 2, 16% for Exam 3 for a total of 46% of the final grade. The final exam then makes up the last 12% of the total grade.

All assignments are due at 11:55 pm the day that the assignment is due. However, students are allowed to submit work a day late with a 12% penalty. After that, students are not allowed to submit for credit. All assignments are submitted through a project

submission server specifically designed for UMD CS courses. Programming projects have release, public, and secret tests attached to them. These tests are the main part of the grade for the projects. The second part of their grade is to manually grade for good programming style.

The course uses Piazza for class communications. The students must follow the same rules that they are to follow in CMSC 131, like not being allowed to post any type of code. The instructor uses piazza for any class announcements. The students are responsible for checking piazza twice a day for announcements.

3.4 Study Participants & Recruitment

Students were recruited from Object-Oriented Programming I (CMSC 131) in Fall 2022, Object-Oriented Programming II (CMSC 132) in Spring 2023, and CMSC 131 in Fall 2023 at the University of Maryland. This study was approved by University of Maryland's Institutional Review Board for human subjects research. At the beginning of each semester, the class was told that their course is part of a study to improve how introductory programming courses are taught. It was made clear that participation is optional and has no bearing on their experience in the course or their final grade. The students were informed that the instructor will not be aware of who is participating and who is not. The students received a paper version of the consent form to sign if they wish to participate.

All courses were taught by the same instructor (discussed in section 3.3.2). However, CMSC 131 in Fall 2023 was also co-taught with another instructor. They co-taught the class by using the same material including programming projects, exams, slides, and sample code. The sections were nearly exactly the same with the only difference being who was actually giving the lecture. The second instructor did also give small breaks to students

to talk to each other about the material like the first instructor, so lectures were also nearly the exact same.

Two teaching assistants (TA) also participated in the study to be interviewed. One TAed for CMSC 132 in Spring of 2023. The other TAed for CMSC 131 in Fall of 2023. Both TAs had above average knowledge of the neurodiverse students. The CMSC 132 TA was diagnosed with ADHD and receives accommodations through the disability services at University of Maryland. The CMSC 131 TA helped to lead student mental health resources on campus in which he provided support to students struggling with their mental health.

Students that participated in CMSC 131 in Fall 2022 were also allowed to participate in CMSC 132 in Spring 2023. Participants had to be 18 years or older and enrolled in CMSC131 in Fall 2022, CMSC 132 in Spring 2023, or CMSC 131 in Fall 2023. While students could be younger than 18 and be taking CMSC 131 or CMSC 132, they needed to be 18 years or older as one of the surveys was the RAADS-R which requires that the person taking the survey to be 18 or older and to give consent to participate in the study.

Recruitment for this study had multiple challenges, specifically in the Spring and Fall of 2023 semesters. The Spring 2023 semester had half the amount of students able to participate than the Fall 2022 and Fall 2023 semester. All interested students were not able to participate due to a breach in the recruitment protocol. All consent forms that were signed after the breach were discarded.

The following tables, Table 3.1, 3.2, 3.3, 3.4, and 3.5 , show the demographics of the students who agreed to participate in this study. Not all participants who signed consent forms completed the demographics survey or the RAADS-R.

Table 3.1 The year of study for all participants that completed the demographics survey

Year	Fall 2022 CMSC 131	Spring 2023 CMSC 132	Fall 2023 CMSC 131	Total
Freshman	90	15	14	119
Sophomore	12	1	3	16
Junior	4	1	2	7
Transfer	3	0	0	0

Table 3.2 The major for all participants that completed the demographics survey

Major	Fall 2022 CMSC 131	Spring 2023 CMSC 132	Fall 2023 CMSC 131	Total
Computer Science	42	10	7	59
Letters & Sciences	35	5	6	46
Computer Engineering	17	1	3	21
Other	13	0	3	16

Table 3.3 The age for all participants that completed the demographics survey

Age	Fall 2022 CMSC 131	Spring 2023 CMSC 132	Fall 2023 CMSC 131	Total
18	86	10	13	109
19	13	6	3	22
20	4	0	1	5
21	3	1	1	5
22	1	0	0	1
23	1	0	0	1

Table 3.4 The gender identity for all participants that completed the demographics survey

Gender	**Fall 2022 CMSC 131**	**Spring 2023 CMSC 132**	**Fall 2023 CMSC 131**	**Total**
Man	53	9	9	71
Woman	50	8	10	68
Non-Binary	4	0	0	4
Genderfluid	1	0	0	1

Table 3.5 The race of all participants that completed the demographics survey

Race	**Fall 2022 CMSC 131**	**Spring 2023 CMSC 132**	**Fall 2023 CMSC 131**	**Total**
African-American/Black	12	2	0	14
Asian	50	6	7	63
Caucasian/White	26	5	7	38
Latino/Hispanic	6	1	0	7
2 or More	12	2	4	18
Other	2	1	1	4

Each student was identified as either neurodivergent traited or neurotypical traited. Students were identified as neurodivergent traited if they self disclosed their neurodiverse condition(s) in the demographic survey or if they scored above a 85 on the RAADS-R. If the student did not disclose any neurodiverse condition(s) or scored below a 86 on the RAADS-R, the student was identified as neurotypical traited. Table 3.6 shows the determined group all participants were assigned.

Table 3.6 The determined neurotype of all participants that completed the demographics survey and/or RAADS-R

Neurotypical/ Neurodivergent	**Fall 2022 CMSC 131**	**Spring 2023 CMSC 132**	**Fall 2023 CMSC 131**	**Total**
Neurotypical	90	26	56	172
Neurodivergent by RAADS-R	22	1	3	26
Neurodivergent by Self Disclosure	30	6	4	40
Total Neurodivergent	52	7	7	66

The students were allowed to disclose their neurodiverse condition(s) in two different ways. They could either say their condition(s) were diagnosed by a medical professional or they self-diagnosed with their condition(s). Self-diagnosis is a controversial topic as supporters argue it helps individuals that cannot get an official diagnosis or were misdiagnosed (Fellowes, 2023). Those opposed claim only medical professionals have expertise to diagnose reliability (Fellowes, 2023). The ability to receive an official diagnosis is a privilege. There are many barriers that affect the ability to be diagnosed. These include not enough professionals for the amount of patients needing appointments, high financial cost, navigating complicated medical systems, struggles with fully explaining life experiences to the professional during appointments, wariness of medical professionals, consequences of an official diagnosis like job prospects and moving to different countries, lack of confidence in the benefit of an official diagnosis, and concerns about medical professionals not diagnosing due to not meeting the diagnostic criteria in the exact way the medical professional is looking for (Fellowes, 2023). Self-diagnosis is considered valid as no one understands the unique experience of the person with a diagnosis like the person that has the diagnosis (Sarrett, 2016).

Self-diagnosis is often misunderstood as a person relating to another individual who is officially diagnosed and then claiming to have the diagnosis the other individual has or using social media to self-diagnose. There are individuals that will 'self-diagnose' based on relating to others or social media; however, self-diagnosis often involves the same criteria that medical professionals are using to diagnose patients like the DSM-V (Fellowes, 2023). Those that believe they have a diagnosis are experiencing symptoms that relate to that disorder, especially commonly understood disorders like anxiety and depression (Rutter et al., 2023). Therefore, in the neurodivergent community self-diagnosis is seen as valid as official diagnoses (Sarrett, 2016).

Self-diagnosed individuals are not using resources that are available for diagnosed individuals. Individuals are only able to access accommodations and services if they have an official diagnosis. Therefore, students may be struggling with neurodiverse symptoms but not be able to address them. Since this study focuses on how to improve introductory programming courses for neurodivergent students, the official diagnosis or particular condition was not important. What was significant for this study was to consider whether the modifications and additions helped students experiencing those symptoms. Hence, self-diagnosed students were categorized as neurodivergent traited. Table 3.7 shows the amount of students that were professionally diagnosed with one or more neurodiverse conditions. Table 3.8 shows the amount of students that were self-diagnosed with one or more neurodiverse conditions.

Table 3.7 The professionally diagnosed conditions of neurodivergent participants that completed the demographics survey

Self-Disclosure (Professional Diagnosis)	**Fall 2022 CMSC 131**	**Spring 2023 CMSC 132**	**Fall 2023 CMSC 131**	**Total**
ADHD	3	1	1	5

ASD	2	0	0	2
Dyslexia	2	0	0	2
Dysgraphia	1	0	0	1
Depressive Disorders	6	2	0	8
Anxiety Disorders	12	2	1	15

Table 3.8 The self-diagnosed conditions of neurodivergent participants that completed the demographics survey

Self-Disclosure (Self-Diagnosis)	**Fall 2022 CMSC 131**	**Spring 2023 CMSC 132**	**Fall 2023 CMSC 131**	**Total**
ADHD	10	0	2	12
ASD	3	0	0	3
Dyslexia	0	0	0	0
Dysgraphia	0	0	0	0
Depressive Disorders	8	1	5	14
Anxiety Disorders	14	2	2	18

In CMSC 131 and CMSC 132, students have various different experiences with programming. They could be learning programming for the first time or they could have had previous classes that taught programming, previously been in extracurriculares that included learning programming, they could have learned programming on their own, ect. It's important to consider the knowledge students are coming into programming courses with. Table 3.9 shows programming languages known to students before CMSC 131 and CMSC 132. Students were allowed to include one or more languages.

Table 3.9 The programming languages known to participants before course began that completed the demographics survey

Programming Languages Known to students before course	**Fall 2022 CMSC 131**	**Spring 2023 CMSC 132**	**Fall 2023 CMSC 131**	**Total**
Java	48	5	7	60
Python	59	14	10	83
C	13	2	0	15
C++	25	3	6	34
C#	4	2	1	7
Javascript	29	4	3	36
HTML	30	2	2	34
CSS	14	2	1	17
Other	21	2	3	26

3.5 Data Sources & Analysis

The data that was collected includes the demographics survey (Appendix D), RAADS-R (Appendix E) results, after-exam reflection survey (Appendix F), grades, interviews (Appendix G), and a modification record for lecture slides. The demographics survey allowed for a better understanding of the student population taking the course. When analyzing the data, the students were divided into two groups: neurodivergent traited students and neurotypical traited students. The students were divided based on the demographics survey and the RAADS-R questionnaire. If a student self-disclosed that they were officially diagnosed or self-diagnosed or scored in the 86-240 of the RAADS-R with a neurodiverse condition, then they were categorized as a neurodivergent traited student. If a student scored in the 0-85 range of the RAADS-R they were categorized as neurotypical traited. The after-exam reflection survey had students reflect on the course after each exam. This included reflections on the material they have learned, their work, and materials given to

them. Grades were collected as they are a useful tool for understanding how well students know the material compared to their peers.

3.5.1 Demographics Survey

The demographics survey was a 24 question survey that would take about 30 minutes to complete. It was administered online through Google Forms. The link to the form was sent to the students by the professor. The purpose of the demographics survey was to better understand the student population that was taking the course. General demographics are important when analyzing student academic performance as gender and ethnicity does affect the probability of getting diagnosed with a neurodiverse condition correctly (Mallipeddi & VanDaalen, 2021). Childhood environment, and previous knowledge also helps to understand how a student performs. A student who has a lot of previous knowledge might do well in class by simply relying on that previous knowledge. Responsibilities outside of school work might affect how a student is able to participate in the course.

3.5.2 RAADS-R

The RAADS-R or Ritvo Autism Asperger Diagnostic Scale–Revised is a peer-reviewed questionnaire that is designed to identify adults who might not have been able to get diagnosed as a child due to underdiagnosis of neurodiverse conditions (Ritvo et al., 2011). It is designed for adults with a normal IQ range and who are above the age of 18. It assesses developmental symptoms relating to diagnostic criteria in the DSM-IV-TR (Ritvo et al., 2011). The questionnaire has 80 statements each with 4 choices: True now and when I was young, True now only, True only when I was younger than 16, and Never true.

The symptoms the RAADS-R screens for fall into the categories of language, social relatedness, sensory-motor, and circumscribed interests (Ritvo et al., 2011). The language category assesses a person’s ability to engage in small talk, likelihood of mimicking words

and phrases from others, and challenges with literal speaking. The social relatedness category evaluates a person's ability to understand the thoughts and feelings of others, likelihood of surrounding oneself with only others that share interests, feelings on being abnormal, bluntness, challenges with knowing when it is appropriate to talk during conversations with others, challenges with group conversations, challenges with object permanence, challenges with maintaining relationships, challenges with body language, likelihood of mimicking others behavior, and likelihood of masking. The sensory-motor category assesses differences in pitch when speaking, differences in volume when speaking, challenges with clumsiness, and challenges with sensory stimuli. The circumscribed interests evaluate preference for focusing on details, anxiety when unexpected events occur, and preference to revolve life around special interests.

The scoring ranges between 0-240. In the process of validating the instrument under controlled conditions, no neurotypical individual scored above a 64 (Ritvo et al., 2011). The RAADS-R was made available online for people to take at home instead of at a psychiatry office. The average score for neurotypical males is 84.2 and for females is 91.6 due to the first question on online tests often asking whether the person taking the test has been diagnosed with autism, suspected autistic, or is not autistic, which skews the results (Engelbrecht & Silvertant, 2020). For the purposes of this study, the 0-85 range was used to define neurotypical traited students and 86-240 for neurodivergent traited students. This means an intentionally conservative scoring scheme was selected in order to underestimate the number of neurodiverse students in the introductory programming course. The students were not prompted about autism before the questionnaire. Therefore, the results would reflect research settings rather than online settings. The results of the RAADS-R were not shared with students. The categorization of the results were only used for the analysis for this study.

This questionnaire does not diagnose autism even if used in a clinical setting. It is used as a tool to better understand a person's experience. If the RAADS-R was used as the only measure for diagnosing autism, there is the possibility of scoring an allistic as autistic and vice versa. The RAADS-R does not distinguish between other psychiatric disorders due to psychiatric conditions having overlapping symptoms (Perinelli & Cloherty, 2023). This study's focus was on neurodiverse students in general instead of those of a particular diagnosis. Thus, using the RAADS-R allowed for a better understanding of which students show more stereotypical neurodivergent traits or more stereotypical neurotypical traits for the students that were not officially diagnosed or self-diagnosed. Not all participants completed the RAADS-R. Table 3.10 shows how many students completed the survey compared to the total number of participants in the study.

Table 3.10 The number of students that took the RAADS-R vs. the total number of participants in the study

	Fall 2022 CMSC 131	**Spring 2023 CMSC 132**	**Fall 2023 CMSC 131**
Number of Students that took the RAADS-R	113	11	21
Total Number of Students Participating in Study	149	33	63

3.5.3 After-Exam Reflection Survey

After each exam, the students were asked to complete an after-exam survey. Having the students take the survey after the exam allowed for students to better reflect on the material they have learned, their work, and materials given to them as they would have to review the course topics to prepare for the exam. The survey was split up into sections: 'Notes, Slides, and Sample code', 'Programming Assignments (Projects)', 'Quizzes and

Exams', and 'Personal Questions'. The questions included Likert scales, multi-select, and short answers. Questions about the 4 sections fell into six categories: Assessment of Material Provided, Knowledge Gained, Assessment of Classwork, Time Management, Mental and Emotional Health, and Personal Perspectives.

The Assessment of the Material Provided category allowed for a better understanding of how students were using the materials provided to them. The questions about Knowledge Gained asked students if they felt they were learning and retaining the material. The Assessment of Classwork questions shed light on how the students felt the material taught related to what they are being graded on. The Time Management questions were created to understand whether the course work demand was possible for all students. The Mental and Emotional Health and Personal Perspectives questions' purpose was to understand how the course was affecting the students outside of the course demands. In Fall 2023, the students had an added question that asked about the added programming project slides.

The after-exam surveys had categorical and short answer questions. There were many questions that were repeated on each of the After-Exam Surveys. The After-Exam Surveys only changed based on the topics taught before the exam. To be sure that a student that took the After-Exam Survey four times was not given greater weight in the analysis than a student who only took one time, the scores for the students that took more than one survey were averaged. Table 3.11 shows The number of students that completed each after-exam survey. The categorical questions were analyzed using five Mann–Whitney U tests were applied to each question. Table 3.12 shows which groups were compared for each categorical question.

Table 3.11 The number of students that completed each after-exam survey

	Fall 2022 CMSC131	**Fall 2023 CMSC 131**

After-Exam Survey 1	42	20
After-Exam Survey 2	26	15
After-Exam Survey 3	19	14
After-Exam Survey 4	13	11
Number of Students that took the Surveys	58	25
Total Number of Students Participating in Study	149	63

Table 3.12 The two groups compared during each Mann-Whitney U Test

Mann–Whitney U Tests	**Group 1**	**Group 2**
Comparison Test 1	All students in Fall 2022	All Students in Fall 2023
Comparison Test 2	Neurotypical treated students Fall 2022	Neurodivergent treated students Fall 2022
Comparison Test 3	Neurotypical treated students Fall 2023	Neurodivergent treated students Fall 2023
Comparison Test 4	Neurotypical treated students Fall 2022	Neurotypical treated students Fall 2023
Comparison Test 5	Neurodivergent treated students Fall 2022	Neurodivergent treated students Fall 2023

The short answer questions were analyzed using open coding (Strauss & Corbin, 1998). The codes were first created by searching for words or short phrases that would answer the question directly without the explanation. For example, the codes for the question *What type of course material do you find is the most helpful? Why?* were recordings, slides, code examples, discussions, website, practice exams, projects, textbooks, piazza, notes, and Javadocs. When codes related to modifications or additions made for the study, those codes were broken into subcodes. For example, the answers that

involved slides were then open coded again into subcodes. These subcodes were information is clear, easy to reference, and easy to read. For the question *Do you feel your programming assignment code reflects your understanding of the material? Why?*, students often wrote yes, sometimes, or no with their answers. However, often the student would write positive and negative things in both the yes and no code. Therefore, the codes that were used were if students said positive things about the programming assignments or negative things about the programming assignments. Table 3.13, 3.14, and 3.15 show the codes and subcodes used for the questions *What type of course material do you find is the most helpful? Why?*, *What type of course material do you find is the least helpful? Why?*, and *Do you feel your programming assignment code reflects your understanding of the material? Why?,* respectively. Each table includes example quotes for each code.

Table 3.13 Codebook for the Question: What type of course material do you find is the most helpful? Why?

Question	**Codes**	**Example Quotes**
What type of course material do you find is the most helpful? Why?	Recordings	"The lecture recordings are most helpful because they allow me to look back at any content I may have missed"
	Slides	"I find the slides to be most helpful because they help to quickly review what was covered in lecture and give a broad overview."
	Code Examples	"Sample code because it helps me to visualize how I should set up my code and write"
	Discussion	"Discussion sections are really useful because they allow us to go through specific code examples in a smaller scale environment where we can ask more questions and engage more

		closely.”
	Website	“All of it was very helpful but I think the website was probably the most helpful resource just because of how effectively it displays all of the information.”
	Practice Exams	“The practice from previous exams is what I find most helpful because I actually get to see what the questions are like and practicing them with the answer key helps me understand better.”
	Projects	“Projects because it lets us practice hands on. As we practice, we gain understanding.”
	Piazza	“Piazza, because the answers on there help to get specific answers to things that still confuse me after slides/example code.”
	Javadocs	“The javadocs during the latest projects; they're the best to work with.”
Subcodes for Slides	Information is clear	“I find that the slides are the most helpful because they clearly list out the rules for every type of function”
	Easy to Reference	“The lesson slides because they refresh my memory.”
	Easy to Read	“slides because they are easy to read”

Table 3.14 Codebook for the Question: What type of course material do you find is the least helpful? Why?

Question	Codes	Example Quotes
What type of course material do you find is the least helpful? Why?	Recordings	"The recordings, because I always come to class"
	Slides	"I find the slides to be the least helpful as they usually do not have a ton of information on them, they cant really show you how to code"
	Code Examples	"The class examples are helpful but I dont completely understand what is happening in them."
	Discussion	"Sometimes discussions feel pointless."
	Website	"The website resources, I find that it's easier to use external resources and guides for any confusion rather than the website"
	Notes	"The written notes because I just don't use them as much as the other resources"
Subcodes for Slides	No Examples	"Slides, because they don't give any real examples."
	Don't Use them	"I often do not look back at slides because just reading the text on the slides is disengaging for me"
	Hard to Follow	"The slides. They are difficult to sort through and organize when reviewing."

Table 3.15 Codebook for the Question: Do you feel your programming assignment code reflects your understanding of the material? Why?

Question	Codes	Example Quotes

Do you feel your programming assignment code reflects your understanding of the material? Why?	Positive	“Yes, because the assignment code actually puts into practice everything I've learned so far.”
	Negative	“I don't understand anything still because I am new to coding. ”
Subcodes for Positive	Implement topics from class	“Yes my programming assignment code implements the bulk key concepts of the material.”
	Improve understanding	“yes I feel like they give me the time to improve my assignments of class topics”
	Shows own ability	“Yes because it shows our own unique thinking process.”
	Lots of Time	“Yes, I finish them in like the first days they open up”
	Fair difficulty	“I do think so. They're fair assignments.”
	Learn what code is doing	“Somewhat, I feel like it helps for us to learn what it is that the code is doing.”
	Simulates coding scenarios in the future	“I think it somewhat does, because it gives us a real life example to apply what we learned in class, which is helpful for simulating coding scenarios in the future.”
Subcodes for Negative	Confused about code	“My own code confuses me sometimes.”
	Too difficult / Structure is bad	“Kind of but sometimes I feel like the programming assignments are too difficult and each programming assignment has a big difficulty leap to the next one so I feel like it isn't steadily getting harder and harder but it's just much

		harder at once.”
	Inconsistency between exams and projects	“For the most part, I think my assignment code does reflect my understanding. However, there is a slight inconsistency between the assignment code and exam code. The pressure of exams may influence my ability to apply my understanding.”
	Missing something	“Not exactly, I feel like I'm missing something in terms of translating understanding into execution.”
	Not enough time	“I do not believe it reflects my understanding. This is because sometimes I do not have enough time to finish projects as I have to juggle school, work, and commuting.”
	Doesn't implement topics from class	“The topics that you don't need in the projects I don't usually understand as well (definitions, types of code that aren't part of projects)”
	Stress	“Sometimes it causes more stress on students to finish the project than to understand the full extent of the concept being taught by the project.”

3.5.4 Grades

Grades are not a perfect measure of how well a student learns material. Traditional teaching and assessment methods penalize neurodiverse students and create demands and workloads that are detrimental to students (Clouder et al., 2020). Both neurodivergent and neurotypical students might get good grades simply because they have already been exposed to the material. Others might do poorly on exams, even though they learned far

more than other students. Students also feel that they learn more if they get better grades, so students' opinions about how well they know the material are often biased (Spooren et al., 2013). Despite their flaws, grades remain important as they are what universities use to assess the knowledge of students and confirm whether or not students have gained knowledge necessary to advance in the program. Therefore, grades should be analyzed when discussing the experiences of neurodivergent and neurotypical students. Grades can be used as a tool to understand how well students know material compared to their peers. In relation to this study, grades were used to help determine if and how the modifications made to course materials helped students.

The grades were analyzed with five t-tests that were applied to each of the assignments, quizzes, and exams that made up the final grade. Table 3.16 shows the two groups compared during each T-Test. Due to point values changing from Fall 2022 to Fall 2023, all grades were converted into percentages to accurately assess the grades the students received.

Table 3.16 The two groups compared during each T-Test

T-Tests	**Group 1**	**Group 2**
Comparison Test 1	All students in Fall 2022	All Students in Fall 2023
Comparison Test 2	Neurotypical treated students Fall 2022	Neurodivergent treated students Fall 2022
Comparison Test 3	Neurotypical treated students Fall 2023	Neurodivergent treated students Fall 2023
Comparison Test 4	Neurotypical treated students Fall 2022	Neurotypical treated students Fall 2023
Comparison Test 5	Neurodivergent treated students Fall 2022	Neurodivergent treated students Fall 2023

3.5.5 Interviews

Semi-structured interviews with students, teaching assistants, and the instructor were conducted in Spring of 2023 (CMSC 132) and Fall of 2023 (CMSC 131) (Appendix G). The instructor was interviewed before the semester began in order to understand how the course was structured. These questions asked about his general experience with teaching introductory programming courses, how he designed the course, and questions that asked about how he would help students that were struggling with different symptoms of neurodivergent conditions. These interview responses were presented previously in section 3.3.2 as a means to better understand the primary instructor and his approach to teaching. No systematic analysis was conducted on the instructor interview as the results did not relate to driving research questions.

Students and TA's were invited to be interviewed after the semester had concluded. All students that signed consent forms were invited to be interviewed through email. The TA's were also contacted through their email. Four students were interviewed in Spring 2023 and Four students were interviewed in Fall 2023. Two of the students in Spring 2023 were identified as neurodivergent traited and two students were identified as neurotypical. One of the students in Fall 2023 was identified as neurodivergent traited and three students were identified as neurotypical. One TA was interviewed in Spring 2023 and One TA was interviewed in the Fall 2023. Both TAs happened to also be neurodivergent traited.

In Spring of 2023, students who had just completed CMSC 132 were asked about their experiences in both CMSC 131 and 132. The students were asked to discuss their feelings towards each of the courses. They also were asked about their personal lives that might have affected their performance in the course (e.g. mental health). The students were asked additional questions about their experiences with the interventions that were introduced in CMSC 132. In Fall of 2023, the students were asked about their experience in

CMSC 131. The students were asked to discuss their feelings towards the courses and about their personal lives that might have affected their performance in the course. The Fall 2023 students had additional questions that asked about the programming projects in more detail to gain insight into if they used the supplementary slides and if so how they used them to help with their projects.

Open coding was used to analyze the interviews with the same codes used for both sets of interviews. Table 3.17 shows the codebook with an explanation of when each code was used. Table 3.18 gives an example of quotes from the interview

Table 3.17 Interview Codebook

Codes	**Meaning**
Prior Experience	Coded when the student discussed previous programming classes or programs.
Choice of Major	Coded when the student discussed why they chose a major in the computer science field
Teaching Staff	Coded when the student discussed the instructor or teaching assistants
Other Students	Coded when the student discussed other students or compared themselves to other students
Students Ability	Coded when the student discussed their own academic ability or standards
Code	Coded when the student discussed the code examples or their own code for programming projects
Projects	Coded when the student discussed the programming projects
Exam	Coded when the student discussed preparation for exams or the exam itself
Class	Coded when the student discussed the lecture or lecture recordings
Discussion	Coded when the student discussed discussion
Student Needs	Coded when the student discussed a need (e.g. more examples)

	that was not met
Stress	Coded when the student discussed stress or negative emotions
Coping Strategy	Coded when the student discussed how they coped with stress or negative emotions
After the Course	Coded when the student discussed how the student used knowledge learned in the course outside of the course
Project Slides	Coded when the student discussed if they used the project slides or how they used the project slides

Table 3.18 Examples for Each Interview Code

Codes	**Example Quotes**
Prior Experience	"This was really the first real programming class I ever took"
Choice of Major	"When I started learning C++ my freshman year of high school, I really got into, I think, it was to build things. And part of that was I wanted to build games."
Teaching Staff	"[Instructor] had this energy that made you want to like, pay attention and also just, like, have a good time."
Other Students	"Most of the students. Had some prior experience in coding so I could tell that they understood everything so it just made it more hard because I was like this is too difficult."
Students Ability	"For me, I learned best by doing, like, I learned a particular, like, method or something like that or particular like objects, I'll implement that into like side projects so I will learn how to use it and where it was useful."
Code	"But I personally found it helpful and I don't think I could really imagine being forced to write stuff on paper and not being able to compile code or edit code or make comments, because another thing I did is that during lecture I would make comments on the code"
Projects	"The project where they didn't give us, like, many, like, they didn't give us clear instructions on what to do."
Exam	"But when you do get to it then 50 minutes fly by quick and so that really stressed me out while I was taking an exam"

Class	"For the computer class I was, like, I wasn't able to write many notes cause it was difficult because it's code. Like for the chemistry and math, you know you like, write solving the problems and all that on the notebook."
Discussion	"The discussion would be 50 minutes, but the materials would go over 20 minutes and then the next 30 minutes would just us sitting in class or people left."
Student Needs	"There's nothing to, like, practice coding, like, you're just learning the information and you have to apply it quickly like there's no middle."
Stress	"I get the actual exam and the exam starts and then I start locking in and, but, then time happens and then I start getting stressed again."
Coping Strategy	"This semester, I did utilize the late grading."
After the Course	"At the end of the course, I felt I was pretty confident of my skills in programming, I mean, and especially using Java language."
Project Slides	"I would always do my project with the slide set and not with the Javadoc or with the descriptions they gave me"

Teaching assistants were asked about their TA experience and their interactions with students during office hours and discussions. They were asked how they would interact with students that exhibit neurodivergent symptoms and their general experiences with students. Coding was not used to analyze these interviews as only quotes could be used due to only one TA participating for Spring of 2023 and Fall of 2024.

3.5.6 Modification Record for Lecture Slides

The modification record documented all changes made to the lecture slides and why the modifications were made. The goal of this data source was to keep a record of how many changes were made as part of revising the course based on the UDL framework. This is important as these logs will help answer the first research question. This also allowed for a reference when updating later portions of the course. Common modifications included fixing grammar and spelling errors, adding white space to the slides for readability, fixing

color issues, and changing font to a more readable font. Table 3.19 shows the modification record for the lecture slides. For each issue, the table gives a reason for the issue needing to be fixed and the change that was made.

Table 3.19 Modification Record for Lecture Slides

Issue	**Reason for Fix**	**Fix**
Duplicate Slide Title	Makes it hard to navigate when titles are the same	Changed slide titles to reflect content on them
Spelling and Grammar Mistakes	Makes text harder to read and understand	Fixed
Not enough white space	Hard to read	Added 6 pt. before bullet and 6 pt. after bullet in the Slide master
Multiple sentences on 1 line	Hard to read	New sentences on new lines
Lists not in list form, separated by commas	Hard to read	Made them lists with bullets
What lists where about was not clear	Hard to read	Bolded list headings
Too much red	Hard to know what is important	Bold and italics some words to reflect what is important
Uses Sans Serif fonts	Sans Serifs are more difficult to read than Sans fonts	Changed font to sans
Use of Purple, Green, and Blue	Not color blind Friendly	Fixed by changing color to Red, Gold, and Blue
Orange text on top of orange background	Not color blind friendly	Changed to not have orange background
Image with text	Cannot be read with a text to speech	Put the text in a text box
Background contrast with text bad	Not ColorBlind Friendly	Fixed the background
Way to much color	Distracting	Made some of the color back to black or blue
Charts unreadable	The charts were really distracting and hard to understand	Changed color, line to dashed, made font larger, Made lines thicker, Change layout

Images not easy to read	Can’t read	Fixed by changing colors and sizes

Chapter 4: Design Findings

This chapter answers the first research question: "How can universal design for learning principles be implemented into an introductory programming course's materials?". In doing so, it presents an analysis of the modification record for lecture slides and discusses how universal design for learning principles was implemented into CMSC 131 course materials. The chapter starts by assessing the course using UDL principles to decide which areas of the course could be improved. Next, the materials that were modified are discussed. Finally, the chapter explains how the lecture slides for the course were updated and how the programming project slides were created.

4.1 Assessment of the Course Using UDL Principles

This section will assess CMSC 131 using Scott et al.'s Universal Design for Instruction Principles for Adult Postsecondary Education: equitable use, flexibility in use, simple and intuitive, perceptible information, tolerance for error, low physical effort, size and space for approach and use, a community of learners, and instructional climate (Scott et al., 2003). These specific UDL principles were chosen due to the principles being created for adult postsecondary education instead of education in general.

This assessment was done for the course during the Fall of 2022 semester before any additions or changes to the material for the study were added. The instructor of CMSC 131 and CMSC 132 created similar formats for in-class material, discussion material, projects, and exams for both courses. The main difference was the topics that are taught, with CMSC 132 presenting more advanced topics that follow on from the content of CMSC 131. Therefore, an assessment of one of the courses would be mostly identical to an assessment of the other. Therefore, CMSC 131 was chosen as it was also the control semester for the study. CMSC 131 was also chosen for this assessment as two semesters of

CMSC 131 were studied. Therefore, the control semester and the experimental semesters could be better compared during analysis.

4.1.1 Equitable Use

Equitable Use refers to instruction designed to be useful and accessible to all students. Students have access to both the sample code used in class and the slides presented in class. This allows students to follow along with the lecture using their own device which can help them pay attention.

All lectures and discussion sections are recorded. Students that cannot be on campus are able to watch the same lecture and discussion sections as those students that are in the classroom. These recordings are made available to the students using Panopto. Panopto has many accessible features that can help students. Panopto has captions built in to help students that have trouble following along with what the instructor is saying. Panopto also allows you to increase or decrease the speed of the video which can help students that get overwhelmed with the speed of the course and those that have trouble paying attention due to the course being too slow.

Students have access to lecture notes made for the course. It's not clear if the notes were made by a TA or previous student of the course. It was made by a past computer science major at UMD. The lecture notes are well organized. The notes are broken up by what topics were learned during what day in 2019. A revision of the notes was done in 2020. The notes have not been updated for every semester, so these notes may not be fully accurate to the current course. These lecture notes are also on GitHub which may be difficult for first time programming students to access and use.

While the lecture slides are accessible to all students in that they all have access to them, improvements to the format of the slides can be made to increase student

comprehension. The lecture slides have many issues regarding accessibility web guidelines. The slides do not make use of white space which can decrease readability ease, as seen in Figure 4.1. Any lists are currently written with commas separating each element and headings are not emphasized, as seen in Figure 4.2. This makes it more difficult for students to access important information quickly. The use of color on the slides can be greatly improved. The contrast of the foreground and background of the slides is not sufficient which makes it harder for students to read, as seen in Figure 4.3. The colors are also not consistent throughout the slides which can make the slides harder to follow, as seen in Figure 4.4. Lastly, the font that is used on the slides is currently a serif font. Serif fonts are harder to read than sans-serif fonts, as seen in Figure 4.5.

Figure 4.1 Slide showing limited use of white space

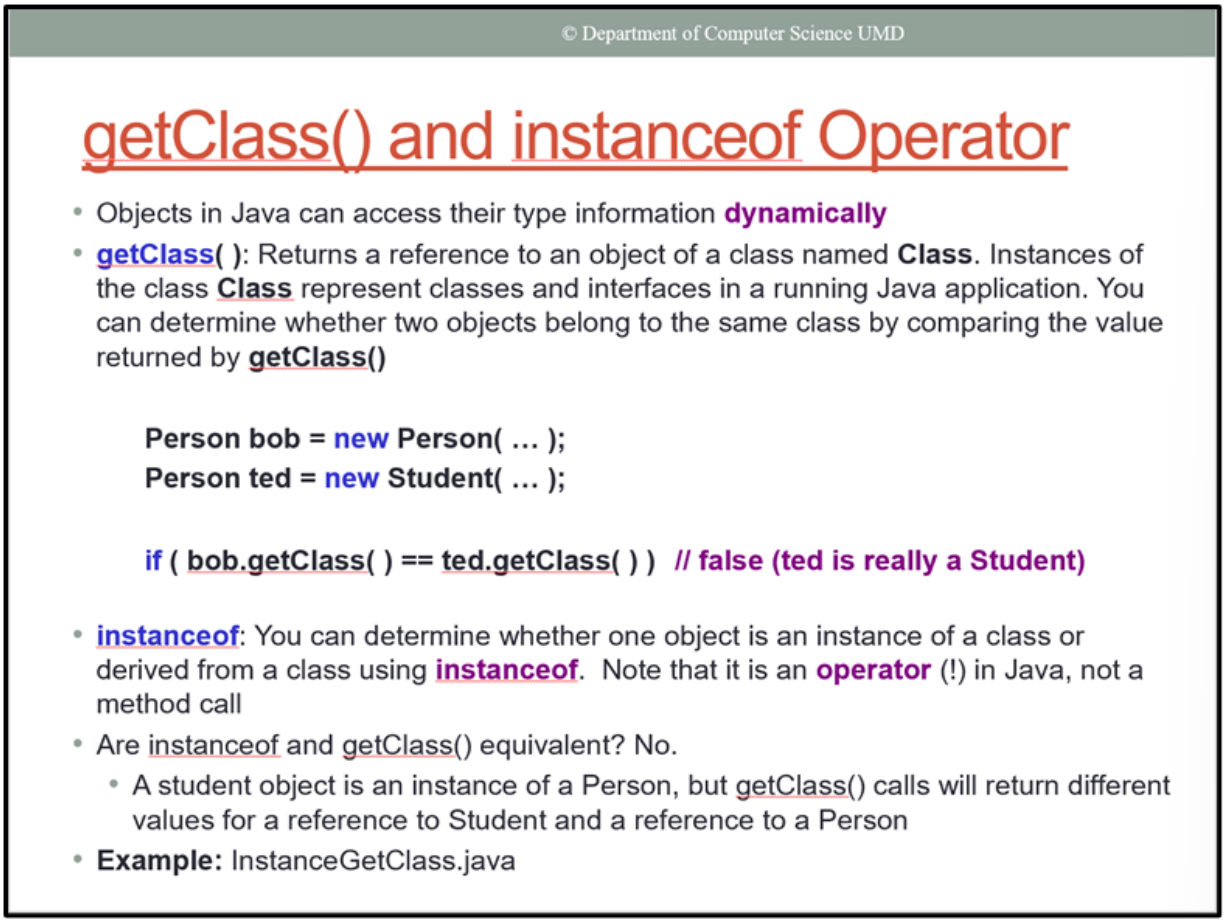


Figure 4.2 Slide showing bad list format

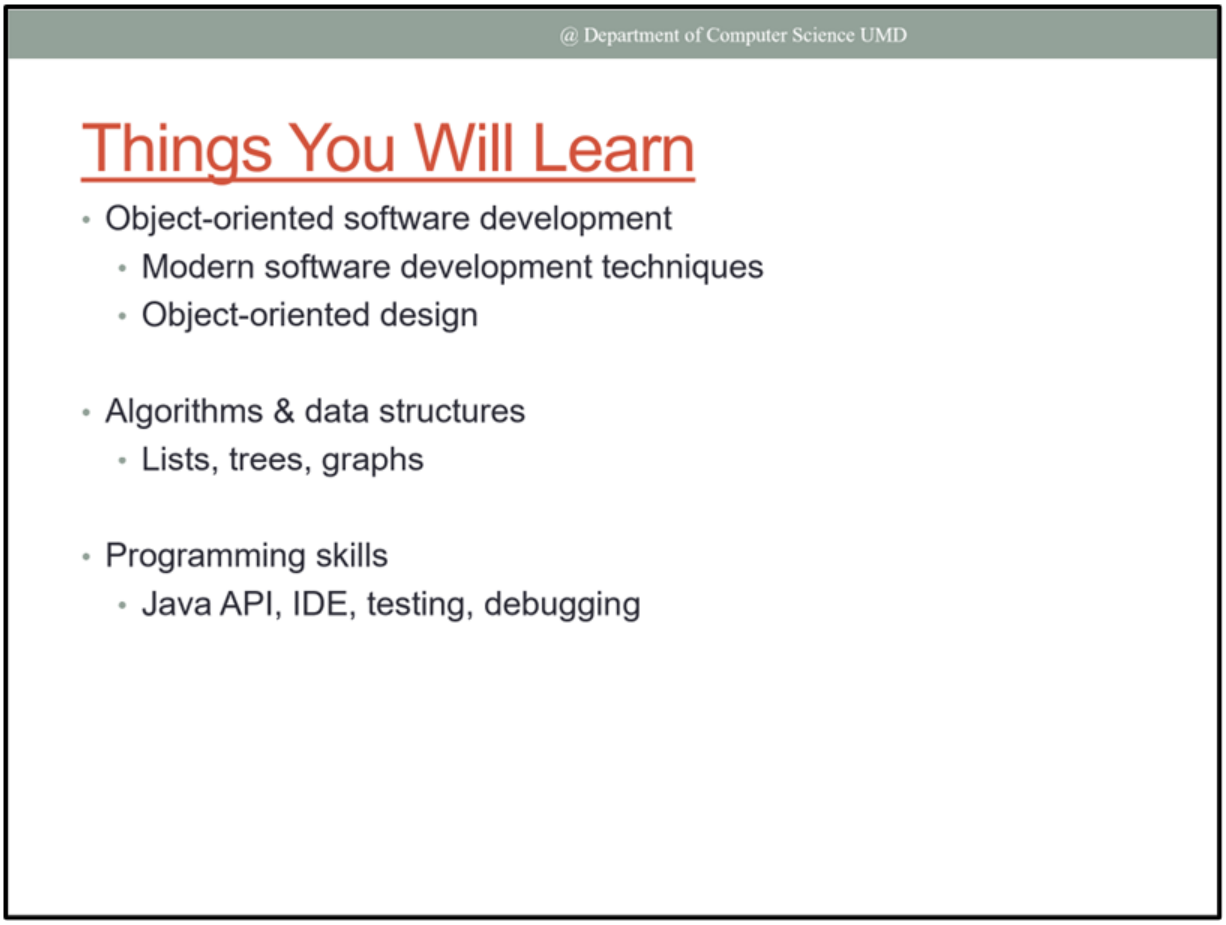


Figure 4.3 Slide showing bad contrast. The last line says 4. d, f.

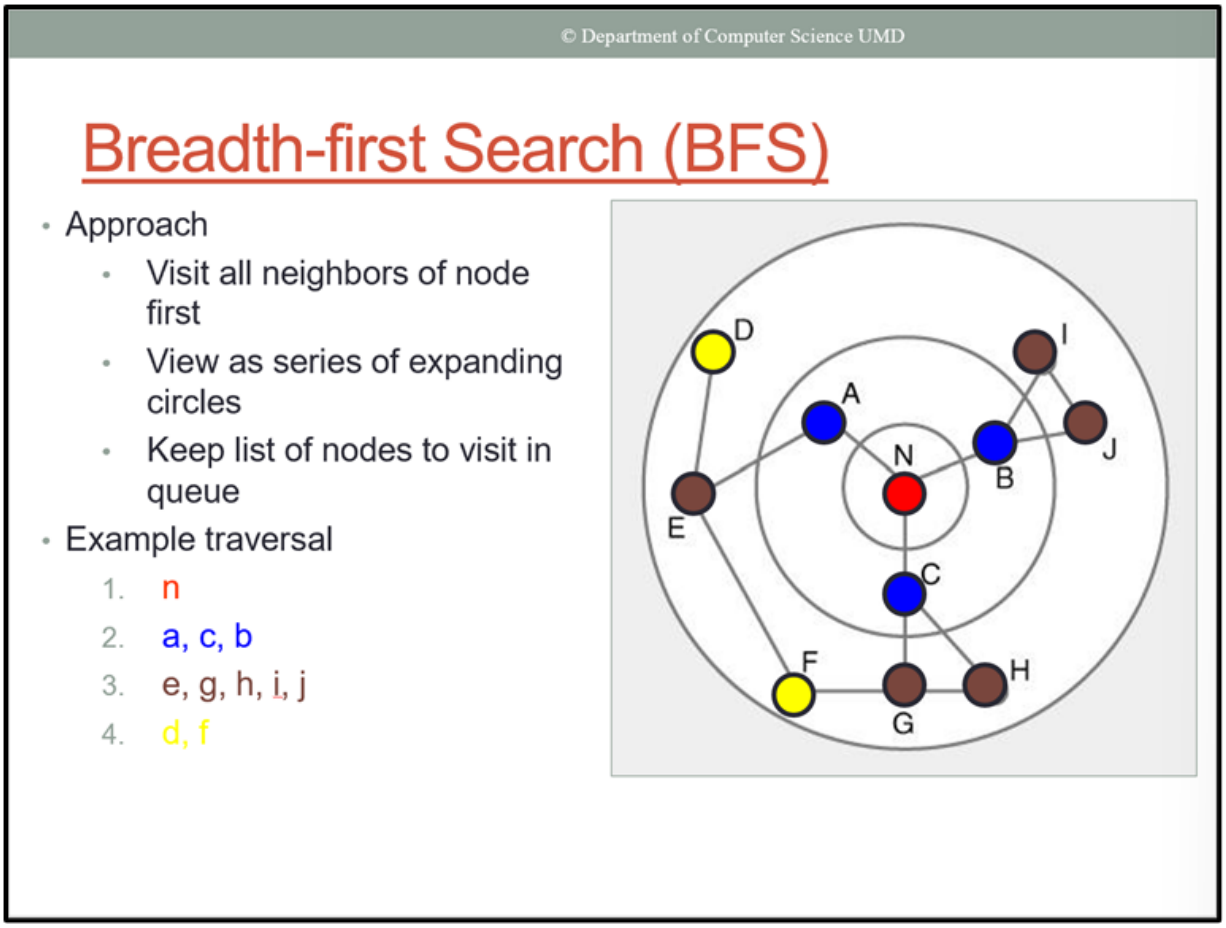


Figure 4.4 Slide showing inconsistent color. Two different red shades are used.

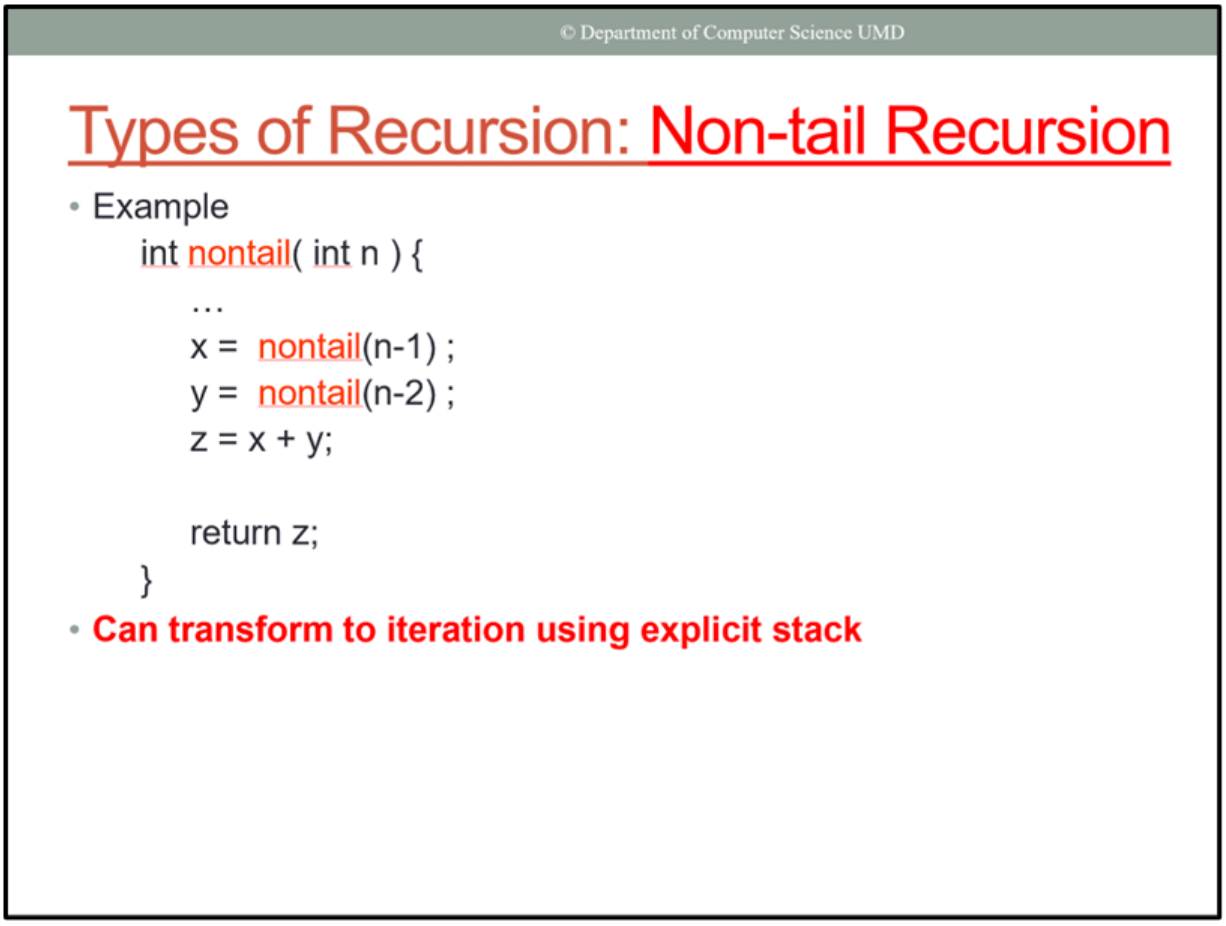
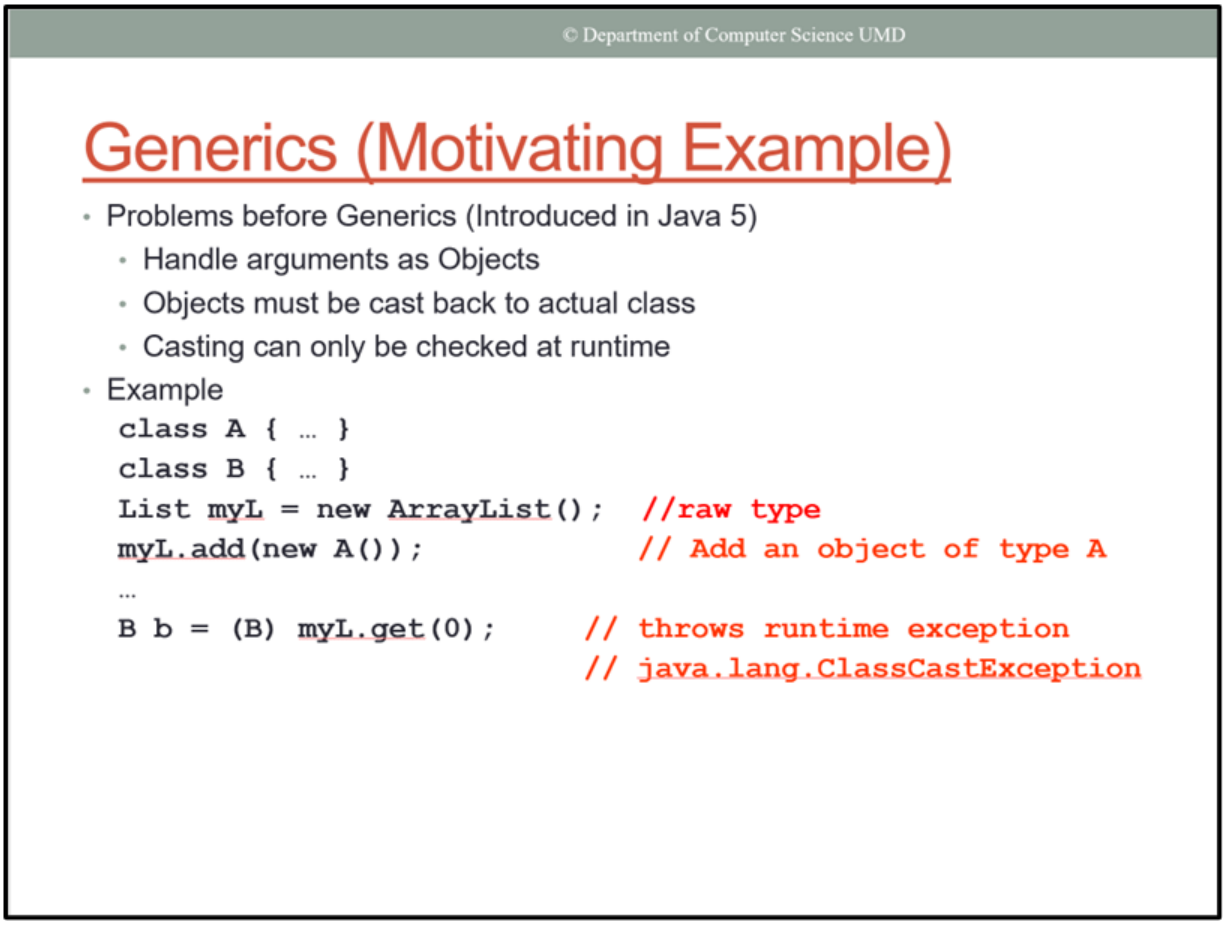


Figure 4.5 Slide showing serif font use. The code is written with a Serif font.



4.1.2 Flexibility in Use

Flexibility in Use refers to accommodating all student abilities and providing them with multiple different methods of learning. The course offers very few ways to learn the material. There is no official textbook for the course. Therefore, the main methods of learning the material is coming to class/watching the lecture and working on the programming projects. Going to class allows for students to ask questions in person. It also helps students to not get behind on lectures like only watching lectures can. Watching the lectures allows for students to speed up the lectures. Students are able to choose whichever method of accessing the lecture material that works best for them. Working on programming projects allows for students to test their knowledge. When the students are

struggling with the projects, it shows them what areas of the material they need to review. They can then ask questions in office hours or on piazza. Both office hours and piazza allow for as long and detailed of a discussion as the student needs.

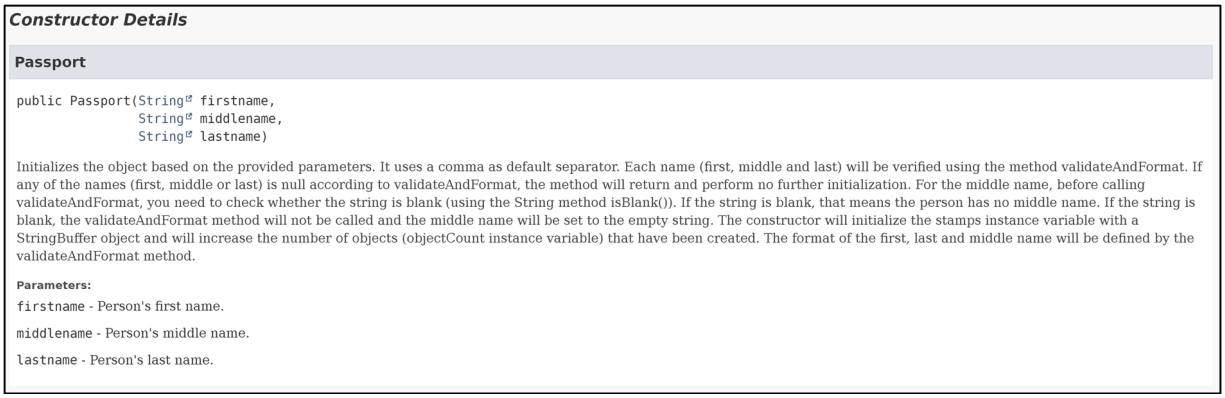
There is also a discussion section that is led by teaching assistants (TAs) and has less students in them compared to lectures. However, this is not required and not fully standardized across discussion sections. The TAs often do not use the full 50 minutes of class time to go over material. TAs often let students out early and also use that time to answer questions related to the programming projects. While using time to answer programming projects questions is helpful to students, often the students are not asking about the material at large. Instead, they are often asking questions about specifics of their particular code which can lead to a shallower understanding of the material.

4.1.3 Simple and Intuitive

Simple and Intuitive refers to teaching in a straightforward way that eliminates as much unnecessary complexity as possible. Programming projects have release and public tests attached to them so students can check if their program is working. However, the students aren't allowed to know any information about these tests. The students are able to ask teaching assistants and the instructor about these tests, but they are not allowed to access more detailed information. They are only aware of if they passed the test or not. Programming projects also sometimes have secret tests. The students are not allowed to know any information about these tests and do not know whether their code passes or fails the test. This can incentivize students to try and continuously adjust their code until they pass tests. They are often just trying to pass the tests rather than learn the material fully. It may seem easier to continuously modify code without understanding why a test is failing rather than figure out where the code is incorrectly computing.

Both the programming projects and programming questions on the exams had similar problems with explanations. These descriptions had large blocks of text that were single spaced as seen in Figure 4.6. This makes it difficult for students to quickly find the specifications needed for the project or question. The projects sometimes did have examples to go with the text, but often the examples just showed the expected input and output. This could make it difficult for students to relate how their code is supposed to compute the output with the input given.

Figure 4.6 Difficult to Read Project Requirements



Constructor Details

Passport

```
public Passport(String firstname,
                String middlename,
                String lastname)
```

Initializes the object based on the provided parameters. It uses a comma as default separator. Each name (first, middle and last) will be verified using the method validateAndFormat. If any of the names (first, middle or last) is null according to validateAndFormat, the method will return and perform no further initialization. For the middle name, before calling validateAndFormat, you need to check whether the string is blank (using the String method isBlank()). If the string is blank, that means the person has no middle name. If the string is blank, the validateAndFormat method will not be called and the middle name will be set to the empty string. The constructor will initialize the stamps instance variable with a StringBuffer object and will increase the number of objects (objectCount instance variable) that have been created. The format of the first, last and middle name will be defined by the validateAndFormat method.

Parameters:

`firstname` - Person's first name.

`middlename` - Person's middle name.

`lastname` - Person's last name.

4.1.4 Perceptible Information

Perceptible Information refers to communicating necessary information, like class announcements, in a way that would be effective for all students. The course website does have a schedule that is updated for when programming projects are due and what material will be taught. The schedule also provides the slides for that lecture week along with the schedule.

However, the course website is not used for class announcements. Instead a separate website, Piazza, is used. By relaying announcements on an online platform, it allows for students that attend lectures and students that do not to receive the same announcements. Piazza is a forum type system. Due to the nature of forums it can be

incredibly challenging to figure out what is necessary and what is not. However, the students are responsible for all announcements. There is an announcement folder on Piazza and Piazza allows students to sign up for emails, but overtime the forum can become overwhelming with the amount of posts. This can often make it easy for students to overlook important information.

4.1.5 Tolerance for Error

Tolerance for Error refers to an instructor anticipating variations in students' learning, pace, and skills during creation of course assignments and assessments. The course does have many programming projects so students are able to not do well on one of their projects without it being detrimental. The programming projects also have release and public tests attached to them in order for the students to check their code. The students are also able to ask about their code in office hours for both the instructor and the TAs for feedback.

However, the programming projects are only about 30% of the grade. The bulk of the grade comes from three semester exams and a final exam. The exams are made up of multiple choice, short answers, and two programming questions. The programming questions are worth about 75% of the exam. If a student struggles with one of the programming questions, they can do very poorly on the exam even if they understand the material asked of them.

There is also not a lot of supplementary practice that the students can work on to help their knowledge. They can experiment with different inputs in the sample code, but there is no structured practice exercise that comes from the course. Since there is also no required textbook, the only option for students is to try and find practice problems outside those that are given to them in the course which can differ from the course content itself.

The exams had many formatting issues which could affect the students ability to show the knowledge they know. The format of the programming problems are large blocks of texts with only some bolding. The exam is also written with a serif font and does not have much space between text. This penalizes students who have a slower reading speed or comprehension speed.

The exams are administered on paper. This means that the students do not have access to an IDE during the exam and all code must be handwritten. By coding on paper, the students do not have access to normal IDE features like feedback on syntax errors and ability to test their code. This also makes it difficult to fix typos. If the students were able to type their homework, students would only have to delete their typo. However on paper if a student makes a mistake, it might make their answer unclear as handwritten work is not as easy to erase.

4.1.6 Low Physical Effort

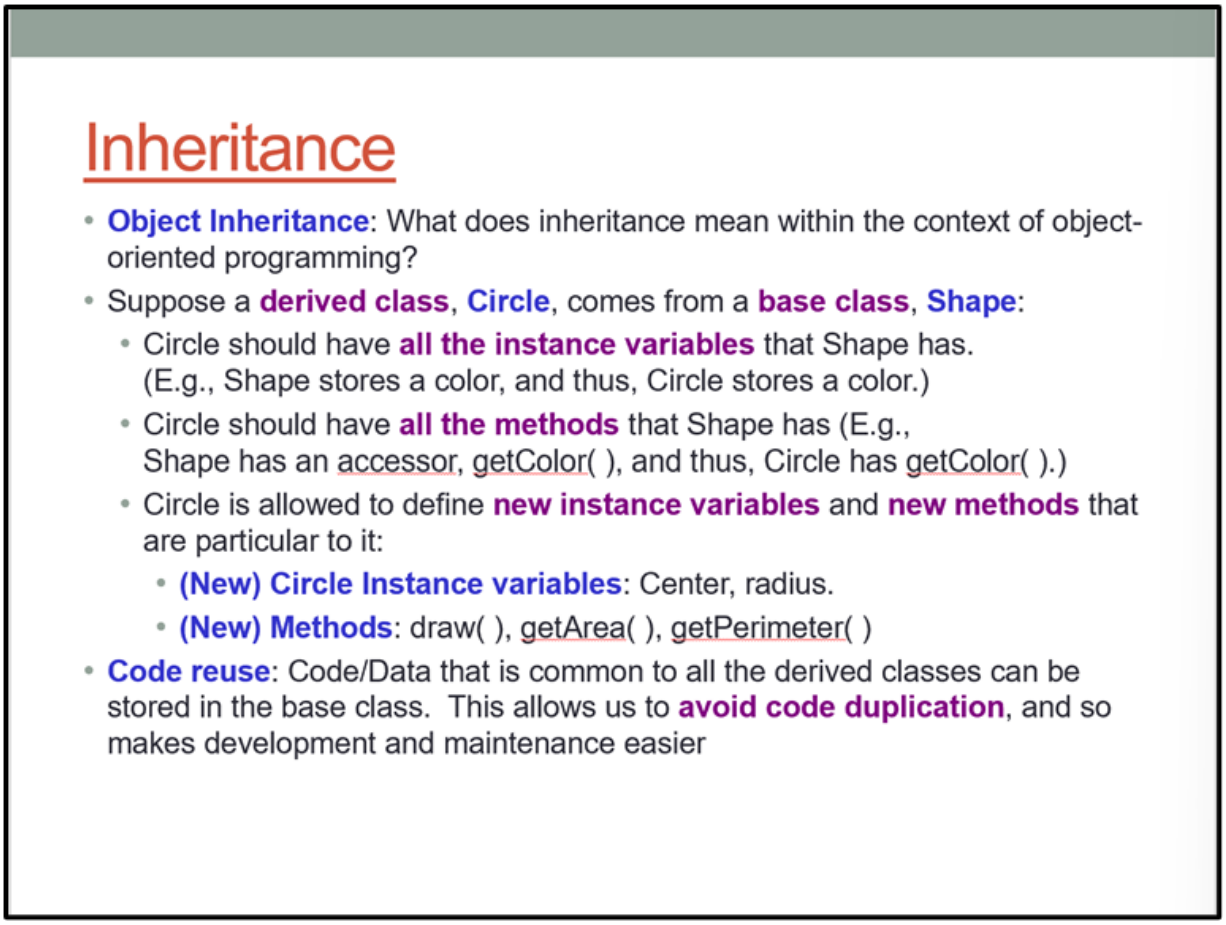
Low Physical Effort refers to designing to minimize nonessential physical effort in order to focus on learning the material. The students use Eclipse as their IDE for the course. Eclipse has an auto code completion feature that allows students to easily use shortcuts. Eclipse gives immediate feedback on syntax errors and has a debug feature to help students with runtime errors.

However, students aren’t able to use eclipse during exams as they are paper based. Coding on paper forces students to have to put in effort to make sure syntax errors and runtime errors do not appear. Paper based exams also have the disadvantage of not being able to move blocks of code easily the way using an IDE allows you. This either forces students to erase possibly a lot of their code or put arrows which are both time consuming and messy.

The lecture slides have many format issues that require students to increase their physical effort as it makes readability and comprehension harder. Therefore, students have to concentrate harder on the slides in order to understand them. This can inhibit their ability to pay attention to the professor. The slides often did not use white space efficiently, as seen in Figure 4.1. The lines of text were single spaced between bullet points which lead to white same on the bottom of the slide and large blocks of text at the top. Furthermore, the text displayed on the slides often were blocks of text with multiple sentences per bullet point. Lists were often listed in a line separated by commas instead of bulleted, as seen in Figure 4.2. Furthermore, the lists often did not have clear headings making the lists hard to navigate. Lastly, the font used for code snippets was a serif font which is harder to read than a sans-serif font, as seen in Figure 4.5.

The lecture slides also had many issues with how they used color to highlight text. One of the more egregious issues was in the form of text not having enough contrast to the background making it hard to read, as seen in Figure 4.3. Another issue was that often colored text would be of the same color hue, but different shades, as seen in Figure 4.4. This made it ambiguous as to whether the different shades gave the importance of the text a different meaning. Lastly, often the colors used on the slides were from the same color family, as seen in Figure 4.7. For example, only cool colors were used for text. By choosing to only use colors from one color family, it lessens the impact of the text on the screen. For students with color deficiency, it makes the content even more difficult to interpret. Therefore, it is harder to tell the difference between text that is blue and purple for example. By choosing colors from other color families, it increases contrast and thus increases the importance of the colored text on the screen.

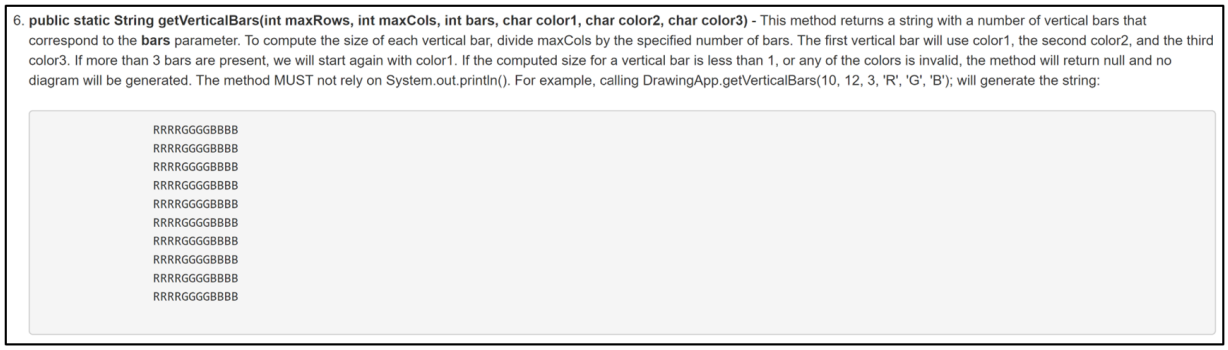
Figure 4.7 Slide showing font highlighted with the same color family



The programming projects had formatting issues that increased the physical effort it would take for students to understand what is being asked of them. Each project had a webpage that just concerned that project. This webpage had large blocks of text with little space between lines of text and bullet points, as seen in Figure 4.8. Some projects did have a Javadoc[2] attached to them to explain methods further, however this then meant that students had to look at two separate web pages to understand what the project was asking of them.

[2] Javadoc is a documentation generator that creates documentation for Java source code.

Figure 4.8 Slide showing serif font use. The code is written with a Serif font.



4.1.7 Size and Space for Approach and Use

Size and Space for Approach and Use refers to designing an environment that is appropriate for all students no matter the student's needs. This is impossible for the instructor to apply at the University of Maryland. The course sizes for the introductory programming courses are large meaning they are assigned to large lecture halls. An instructor simply cannot change the environment to meet all students' needs. The teaching assistants possibly could apply this principle, however all of the tables in the computer science buildings are attached to power sources. Therefore, tables cannot be moved. However, the small environment of discussion does allow students to move to different seats easily.

4.1.8 A Community of Learners

A Community of Learners refers to an environment that promotes communication among students and between students and instructors. The instructor facilitates communication with students by building breaks into the lectures so students can discuss with other students what they are learning. Piazza also is a great tool for student interaction as the forum style allows students to have long discussions. The students are also allowed

to post anonymously, so if they feel uncomfortable asking a question they are able to ask it without fear of judgment.

The students are allowed to discuss the first few programming projects with each other, but after that they aren't allowed to. The students aren't allowed to look at each other's code or post their code on piazza for help. Therefore, students aren't really able to help each other with programming projects and have to seek help from only TAs or the instructor.

4.1.9 Instructional Climate

Instructional Climate refers to designing the course to be welcoming and inclusive. The course does not have any policies or forms of instruction designed to specifically be welcoming and inclusive. The instructor does encourage students to talk to others and go to office hours which can help facilitate this environment. Even though this is an introductory programming course, the computer science major is incredibly competitive which can often make students feel that they don't belong in the major because they compare themselves to others. The instructor does try to encourage students to not compare themselves to others, but this is a larger departmental climate issue, so encouragement can't fully fix the problem. Most computer science courses at UMD are graded with a curve. While CMSC 131 is not, the curve culture still affects how the students perceive themselves and others in the department.

4.10 Summary of Assessment

Equitable use was used properly when the sample code, slides, and lecture notes were made available to students. However, improvements to the format of the slides and the lecture notes being updated for every semester would increase the usability and accessibility for students.

The different methods of learning which follow the flexibility in use principle are coming to class, watching the lectures, going to office hours, using piazza, and going to discussion sections. Flexibility in use was applied when allowing students to come to class or watch lectures and going to office hours or using piazza for questions. However, improvements can be made to discussion to allow for another method of learning that is in a smaller setting by fully utilizing all of the discussion time. Introducing a recommended textbook can also be helpful to students that like reading as their main method of teaching.

The programming projects and programming questions on exams do not follow the simple and intuitive principle. The release, public, and secret tests for the projects add complexity to understanding if code is working properly as students are not allowed to ask for detailed information about each of the tests. This can make it more confusing for students who are trying to understand why their code is failing. The programming projects and programming questions on exams requirements also can be improved. The format of these specifications make it difficult for students to quickly find what is expected of them.

The perceptible information principle is used correctly when communicating the schedule of the course being taught. However, improvements can be made to the way necessary information is relayed. Since piazza is a forum type system, important information can become buried in other posts making it difficult for students to be responsible for all announcements.

The course allows for tolerance of error by having many programming projects. The students are also able to see if they are passing the release and public tests. This allows them to check their code and ask for help if their code is not passing tests. Tolerance for error is not used with exams as 4 exams make up 70% of the grade. This puts students who are bad at test taking at a significant disadvantage. The exams also are administered on paper which makes it difficult to correct during the exam.

Low Physical Effort is applied correctly by having students use an IDE with auto code completion and immediate feedback on syntax errors. However, they are not allowed to use Eclipse during exams as they are paper based. The format of the lecture slides and programming projects can be improved to increase readability.

While not the fault of the design of the course, large lecture halls and unchangeable discussion rooms do not follow the size and space for approach and use principle.

The community of learners principle is applied by building breaks into lectures for students to discuss material with each other. It is also used when having piazza be a tool used for students to have long discussions with each other. Allowing students to work together on projects can increase the use of the community of learners principle.

While the instructional climate principle isn't purposely built into the course, the instructor encourages students to talk to each other. While not the fault of the design of the course, the department has a climate issue that creates unnecessary competition which often affects the incoming students.

4.2 Materials to be Modified

It was important to not completely change the policies and material that the instructor uses to teach, so the instructor continues to feel comfortable and in control of their course. After assessing the course using UDL principles, the three areas of the course that were identified for improvements were the lecture slides, programming projects, and exams. The previous lecture slides format opposed equitable use and low physical effort. The current format of the programming projects went against the simple and intuitive, low physical effort, and a community of learners principles. The exam format opposed equitable use and low physical effort. The current format of the exams went against the simple and

intuitive, low physical effort, and tolerance for error principles. However, any improvements to the course were discussed with the instructor of the course.

The instructor was comfortable with changes to the lecture slides as long as content was not changed. Therefore, the modifications of the lecture slides involved modifications of format. The instructor was also comfortable with the creation of new material to the projects to help students understand what the project is asking of them. These project slides would walk students through the projects without explicitly showing the student how to write the code. While the control students only received dense written descriptions with limited examples, the Fall 2023 students were also given access to new supplementary project slides with a more readable format of the descriptions and more examples.

The professor was not comfortable with any changes being made to the exams. Improvements to the exams that could have been made would be allowing students to take the exam using an Integrated Development Environment (IDE), changing the exam questions wording and format, and allowing students more time to take their exam.

4.3 Lecture Slides

Before any modifications, the lecture slides were split by topic where one topic that has more content could be split between two different slide packets. The slide packets were then split up by week with each week consisting of 1 to 5 different slide packets depending on the density of the topics being taught. The organization and content of the slides were not changed in the modification. The slide packets were created by the instructor of the course. However, some slide packets were based on material provided by other members of the computer science department. Even though all slides were created using Microsoft PowerPoint, the slides would be exported to pdf form to be presented in class. They were also provided to students in pdf format.

To increase the UDL principle of equitable use, modifications needed to be made to make the lecture slides more useful and accessible to students. Low physical effort was also applied when deciding what modification would increase the readability of the slides. These modifications to the lecture slides began by using built-in accessibility tools like Microsoft Powerpoint's Accessibility Checker, Spelling and Grammar Checker, and the Flesch Reading Ease test. Manual modifications were first guided by the Web Content Accessibility Guidelines (WCAG) 2.1 (*Web Content Accessibility Guidelines (WCAG) 2.1*, 2023). These guidelines were chosen as all course material was made and presented using a computer. Therefore, all design recommendations for a webpage could be translated for a slide packet. Since this study focused on neurodivergent students, less emphasis was put on guidelines for physical disabilities like blindness and deafness. The changes made to the lecture slides included adding white space to the lecture slides, changes to the format of lists, changes to the use of color on the slides, and changes to the font used.

More emphasis was put on cognitive accessibility guidelines that would improve the experiences of neurodivergent students like readability. Not all changes made were explicitly due to the WCAG. The choice of the type of font, Sans-Serif, was guided by recommendations from dyslexic associations. Instead of Serif fonts like Times New Roman, Sans-Serif fonts like Verdana, the font used to write this dissertation, are suggested as Sans-Serifs are easier to read (Rello & Baeza-Yates, 2016). The choice of color for text was decided based on color theory and visibility. Format text using bolding and italicizing was done sparsely, but also used to emphasize important sections of the material on the slide like headings and instructions. In the following sections, additional details about how the changes were made are discussed and images showing the changes are presented.

4.3.1 Accessibility Checker

Since all lecture slides were created using Microsoft PowerPoint, every slide packet was checked using the Accessibility Checker built into the software. Most of the inspection results were missing object descriptions, check reading order, and duplicate slide title. The missing object descriptions and check reading order errors relate to how text to speech will not be able to read alternative text for images on the slides and text to speech might not read the text on the screen in the order expected. These errors were manually fixed when needed. The last was not an accessibility error, but a warning. This warning was about how slides often had duplicate slide titles. This is an accessibility error due to navigation of the slides being more difficult. This was handled by adding '(cont.)' to the slide title so students would be able to search for what they are looking for faster. When modifications require slides to be split, '(cont.)' was also used.

4.3.2 Spelling and Grammar Checker

Microsoft PowerPoint's built-in spelling and grammar checker was then used. There was often only the mistake of including having double spaces in a sentence when it is not needed. For example, '*In Eclipse we define them as we define classes*' was changed to '*In Eclipse we define them as we define classes*'. However a few spelling mistakes were also fixed. For example, '*The class schedule is available in the Schedule section of thee class web page*' was changed to '*The class schedule is available in the Schedule section of the class web page*'. Spellcheck had to be used carefully since code often has words that are not recognized by the software.

4.3.3 Flesch Reading Ease

It is important for students to easily understand what the material on the slides are saying so that they can focus on learning the material (Initiative (WAI), 2024e). Therefore,

the vocabulary not related to the material being taught should be as simple as possible. In order to check the readability of the actual sentences on the slides, the Flesch Reading Ease test was used. The Flesch Reading Ease test is built into many different applications. It calculates readability on a scale from 1-100 where higher scores mean the material is easier to read (Boutemen & Miller, 2023). The score is based on the number of syllables per word and the number of words per sentence. While this is not a perfect metric for most applications, lecture slides need to be concise so that important information would be highlighted. Therefore, the Flesch Reading Ease test was a good measure to check how easy the slide would be to read by the students during lecture. Each slide packet was checked with the aim to score above a 70 on all slides as a score of 60-70 is considered to be the standard recommended readability level (Boutemen & Miller, 2023). All slides did have a readability level above 70, so changes to the material focused on the format of the slides.

4.3.4 White Space

Per the WCAG 2.1 guideline 'Use White Spacing', white space should be put around objects and text so each section is clearly separated (Initiative (WAI), 2024a). Using white space helps students to read material and comprehend the most important parts of the material. Dense blocks of text decreases comprehension. Increasing the white spaces on slides will help dyslexic students to read and comprehend material more efficiently. This can also help Autistic students understand what is important on each slide.

In order to add white space to each of the lecture slides, 6 pt. spacing was added to main blocks of text in the slide master before and after each line of text. Often this meant, one slide had to become two. In most cases the slides were split before an example, as seen in Figure 4.9 or before another definition.

Original Slide **Modified Slides**

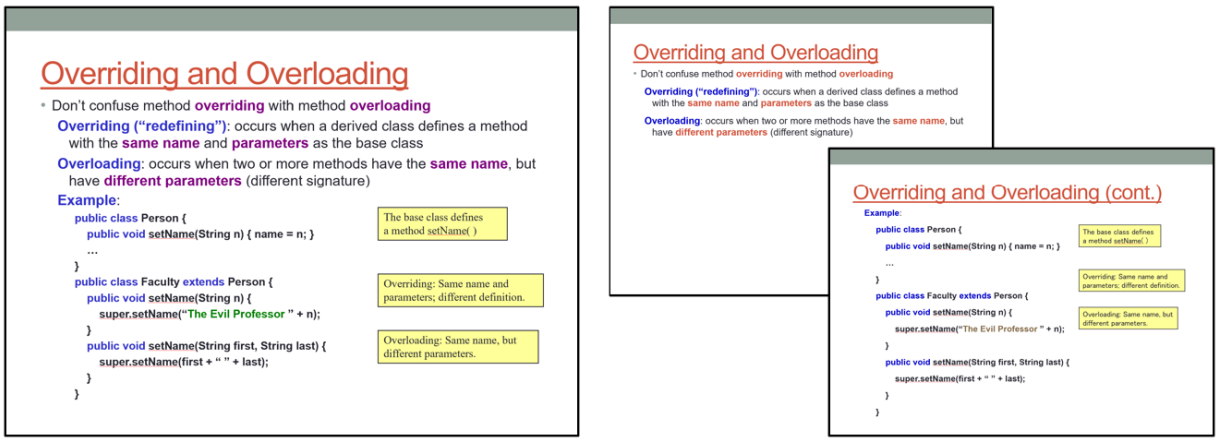


Figure 4.9
A comparison of the original slide and the modified slide with the added white space

Another way white space was added is to have only one sentence per bullet. Often, the slides had multiple sentences for one bullet point which made analyzing what content was important difficult as seen in Figure 4.10.

Original Slide **Modified Slide**

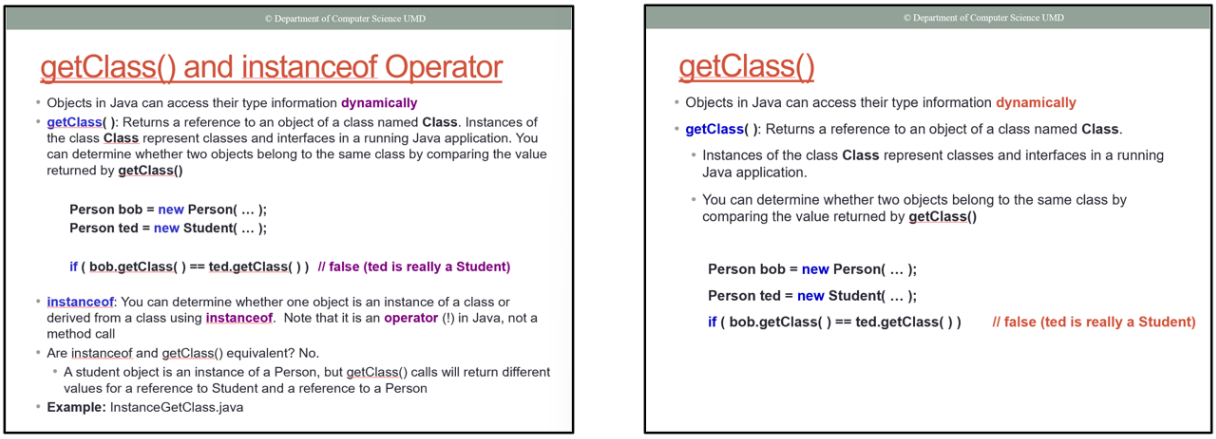


Figure 4.10
A comparison of the original slide and the modified slide with only one sentence per bullet

4.3.5 Lists

Per the WCAG 2.1 guideline 'Keep Text Succinct', text should be kept as succinct as possible (Initiative (WAI), 2024c). Often lists were written on one line separated by commas as seen in Figure 4.11. By changing the list to a bulleted list, the list becomes easier to read and understand. Another issue with the original list was that the lists often did not have clear headings. The same guideline advises that short descriptive headings should be used above the lists. Therefore, every list was preceded by a heading so that students can quickly see what the list pertains to as seen in Figure 4.11. This helps dyslexic students to read and comprehend material quicker. This can also help students with ADHD as it increases processing speed which will help students to easily pick up material if they are distracted.

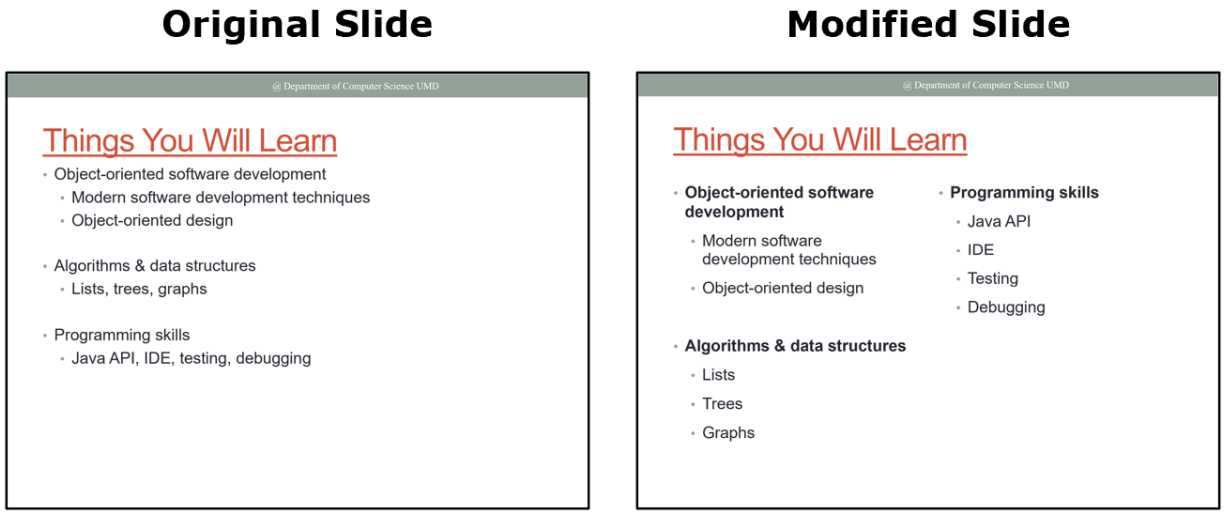


Figure 4.11
A comparison of the original slide and the modified slide with bullet points instead of commas and more prominent headers

4.3.6 Color

There were many improvements that could be made in regards to the color used on the slides. Per the WCAG 2.1 guideline 'Ensure Foreground Content is not Obscured by Background' (Initiative (WAI), 2024b), it is a necessity for foreground content to not blend into the background. This means that there must be a strong color contrast between the background and any text that is used. A few times the color chosen for text was illegible on a white background as seen in Figure 4.12.

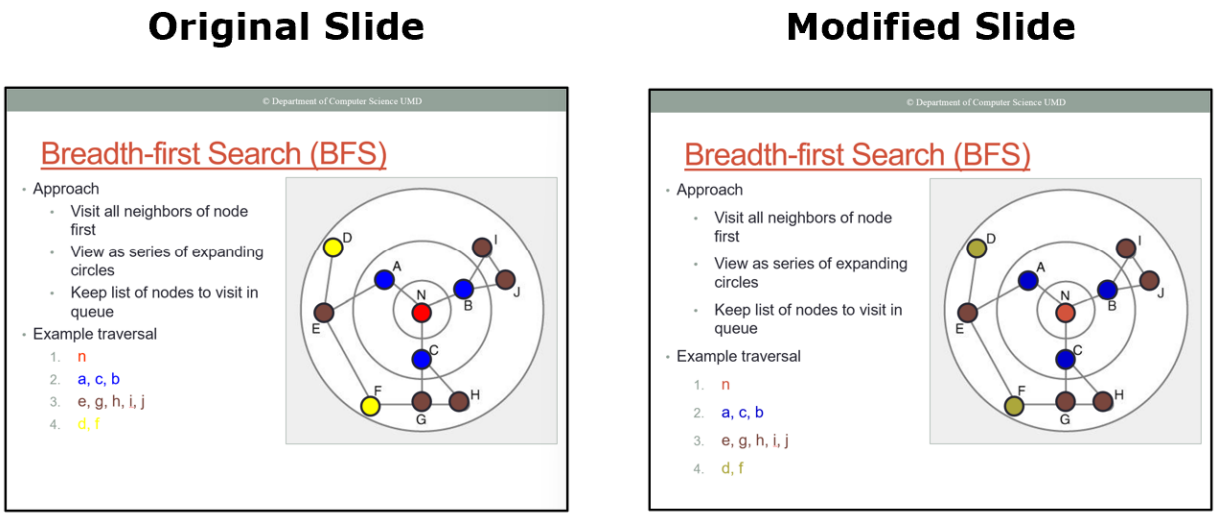


Figure 4.12
A comparison of the original slide and the modified slide with unreadable color used. In the original slide, the last line of text 4. d,f is unreadable due to the contrast of yellow text on a white background not being high enough.

When considering that guideline, the decision to change the use of blue and purple to highlight important text was made. Instead the use of colors that had more of an intense difference like red and blue were used as seen in Figure 4.13. Blue and purple are analogous colors, therefore, when looking quickly through slides it can be difficult to distinguish between the two colors.

Original Slide **Modified Slide**

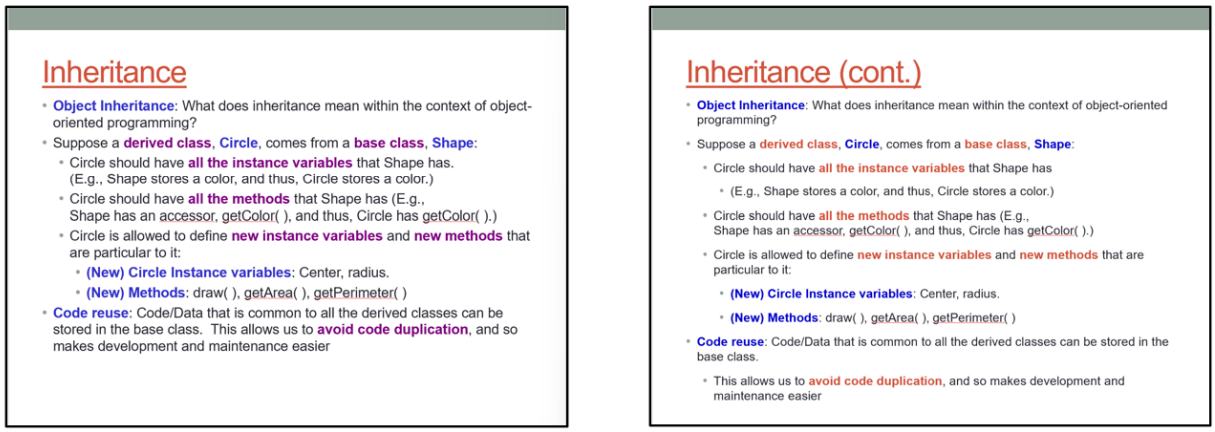


Figure 4.13
A comparison of the original slide and the modified slide replacing the purple text with red

Per the WCAG 2.1 guideline 'Use a Consistent Visual Design', visual design needs to be consistent throughout the slide packet (Initiative (WAI), 2024d). This helps students to quickly recognize elements. The slides used text colors that were of the same color hue, but were different shades. This could make it unclear to students whether the different colors are meant to highlight important information or not as seen in Figure 4.14. In order to make this less ambiguous, all colors were made the same if they were in the same color hue. Having slightly different shades of the same color hue also puts color blind students at a disadvantage. They might not be able to see the slight difference in shade. If the slight shade difference was meant to highlight important information, these students would not be able to see this.

Original Slide **Modified Slide**

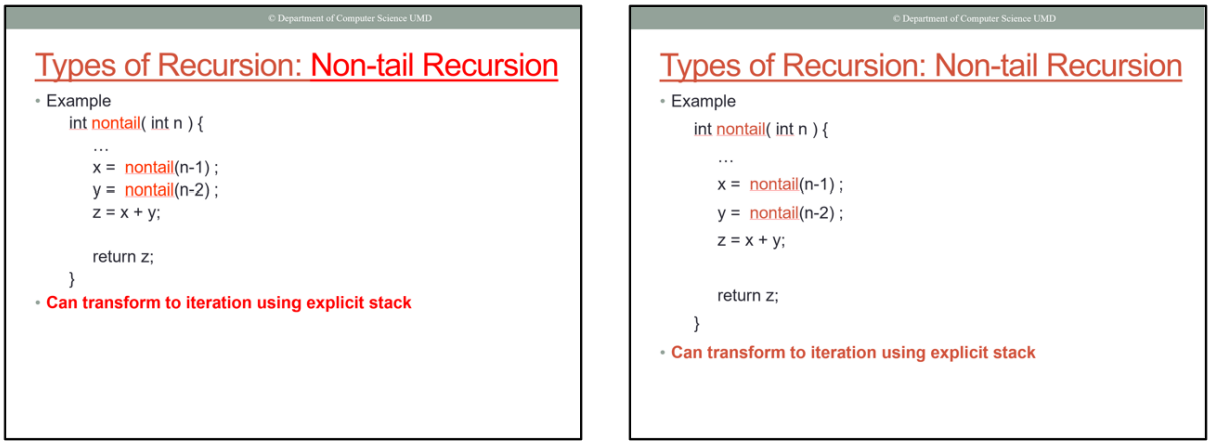


Figure 4.14
A comparison of the original slide and the modified slide replacing text with the same hue with the same color

In some cases, there was too much use of the same color. This often made it difficult to understand what was important information. In this case, italics and bolding were used to emphasize different parts of the information to separate the information as seen in Figure 4.15.

Original Slide **Modified Slide**

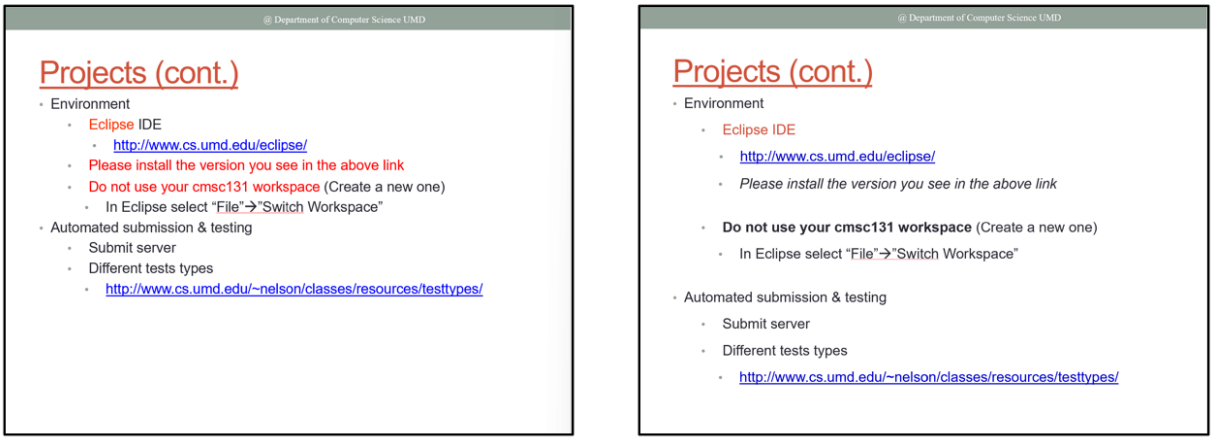


Figure 4.15
A comparison of the original slide and the modified slide replacing color with bolding and

italics

4.3.7 Font

While font choice is not a requirement for WCAG 2.1, dyslexic and low vision associations recommend the use of sans-serif fonts like Arial, Comic Sans, and Verdana (Rello & Baeza-Yates, 2016). These fonts are also beneficial for those without dyslexia (Rello & Baeza-Yates, 2016). While most of the content on the slides were written in a sans-serif font, often code was written in a serif font. While it is appropriate to differentiate code from the rest of the text on the document, the font might make it difficult for dyslexic students to read. MS PGothic was chosen to be the font that replaced the Courier New font the code was originally written in as seen in Figure 4.16. This new font was visually different from the Arial font that the rest of the material was written in but still a sans-serif font to help with readability.

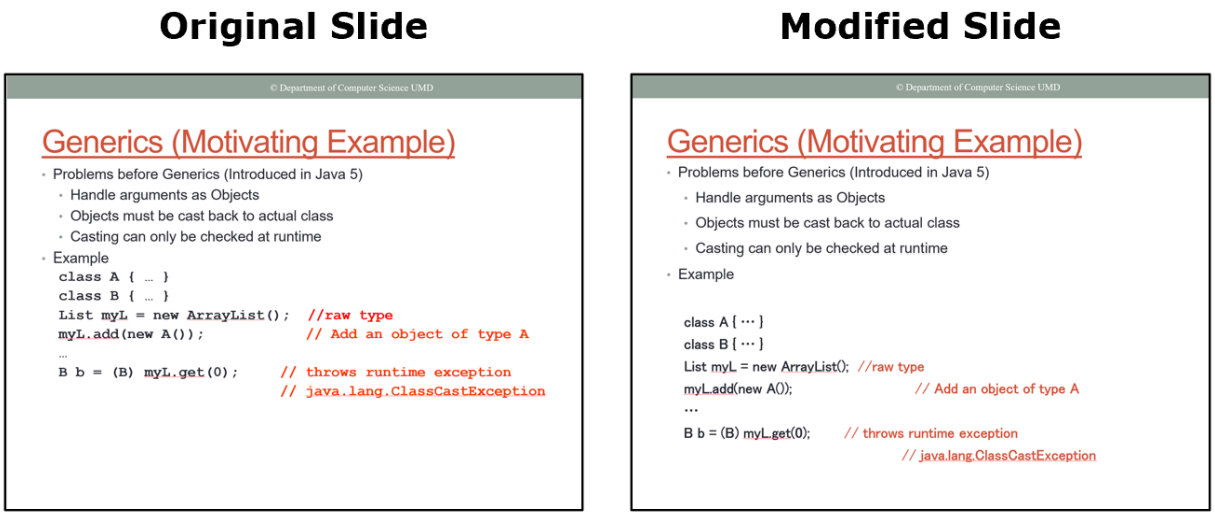


Figure 4.16
A comparison of the original slide and the modified slide replacing serif font with sans-serif

4.4 Programming Projects and Project Slides

Each programming project had a webpage that discussed the objective, the overview, grading, code distribution, and specifications of the project. The objective was a brief description that explained what part of the material the student would be practicing by doing the project. The overview gave the students a brief explanation of the project. The grading section detailed the rubric for the project. The rubric often consisted of public tests, release tests, secret tests, and style. Sometimes it would also include student tests. The specifications for the project would give an explanation for each program, method, or constructor that the student was expected to write. The specifications might have the student also look at a Javadoc to explain the method or constructor expected for each class. These specifications would sometimes include examples. If the project required students to create a class where multiple methods depend on each other, a sample driver might be supplied instead of a single example. The webpage also would include other information that students would be expected to know like announcements, how to submit, due dates, whether the project can be discussed with other students.

For each project, the instructor provides a zip file that contains the code distribution. The project's code distribution contains a programs package folder that provides students with a outline for the programs that the students need to implement, a package folder of public tests that the students can use in order to check their code, a folder that contains the expected results of the public tests, and a folder that would contain the results that are generated from the students code when they run their public tests.

Computer Science classes that teach programming at UMD use three tests for class projects. These are public, release, and secret tests. When a student submits their code to the submit server, the public, release, and secret tests associated with the project are run. The test will automatically either pass or fail a student's code. A majority of the student's

grades for the project are calculated with these tests. A public test is a test that students have access to. They know both what the test is and what the expected result is. These tests are provided with the project's code distribution.

Release tests are tests that have no information provided with them. Students do not have access to the information that describes the test and the expected results. The release tests are meant to verify the functionality of the students code, but they are only allowed to be seen a set number of times. The system gives the students tokens. The instructor will specify a duration of time that the students can use the tokens. For example if an instructor allows a project to have 3 tokens and a time duration of 24 hours, then a student has to wait 24 hours before they would be able to use another token. Therefore, it would take the student 72 hours before they could use all three tokens. This is to help encourage students to start projects early, so they can make sure to test as many times as they can if they need to. These release tests are used as part of the students project grade. Most of the projects require students to pass all of the public tests before they can test their code with release tests. Secret tests are tests that the students do not have any access to. They do not know the information that describes the tests or the expected results. They only know whether they passed the tests after they received the grade for their project.

Students are allowed to ask TAs in office hours about both public and release tests. However, TAs are allowed to help the most when students are struggling with their code passing the public tests. Because the students can see the specifications and the expected results, the TAs are able to go through the entire test with the student to show how the test is testing their code. For release tests, students are allowed to talk to TAs about which ones they are failing, but the TAs need to be more vague. They can only give hints as to what the student should try instead of going through the code and test with the student. The TAs are not allowed to discuss secret tests at all until the students have received their grades. The

TAs are also allowed to discuss the release tests in more detail once their grades have been released.

Most of the projects are also graded by the TAs manually for style. The students are given Style Guidelines on the course website that they have to follow when writing their code. The TA's grade for style based on if the students are adhering to the following rules:

- Variable names being descriptive and camelCased
- Consistent curly bracket placement
- Using braces around loops and conditions
- Consistent indentation
- Spaces between binary operators
- Defining variable close to where they are being used
- Define constants, like π, when needed
- Avoid long lines of code
- Reduce the number of lines by defining variables of the same type in the same line
- Do not over use break or continue statements
- Do not call static methods with class instances
- Remove variables that are never used, unnecessary import statements, and code duplication
- Clarify code with empty lines and comments
- Methods should have descriptions of what it is doing
- Simplify code as much as possible
- Make code neat, well organized, and consistent

During Fall of 2022, the students were expected to complete 8 projects: Java Basics-Conditionals, Java Loops, Drawing App, Passport Class, PhotoManager, Array Utilities, Diagram System, and Media Rental Manager. In Fall of 2023, a new project Photo

Processing System was introduced. Due to consistency and timing, project slides for only the 8 projects that were assigned in Fall 2022 were created.

The Java Basics/Conditionals project had students practicing writing Java's expressions, working with input and output, using conditions, and becoming familiar with the course tools that would be used throughout the course like the submit server. The Java Loops project gave students the ability to experiment with Java's loops. The Drawing App project gave students an opportunity to work on writing Java's static methods and nested loops. The Passport Class had students practice basic class definitions.

The Photo Processing System projects gave students the opportunity to practice writing classes, using Java's 'this' keyword, using exceptions, using the StringBuffer class, and parsing strings. The PhotoManager project had students practice using class definitions, using classes from another class, and using the ArrayLists and Collections classes. The Array Utilities project had students experimenting with one-dimensional arrays. The Diagram System project had student work with two-dimensional arrays and interfaces. The Media Rental Manager project allowed students to practice designing a system. All classes and interfaces were designed by the students.

Table 4.1 CMSC 131 Projects

Fall 2022 CMSC 131	**Fall 2023 CMSC 131**	**Project**	**Objective**
P1	P1	Java Basics/Conditionals	Practice Java's expressions, input/output, conditionals, and course tools
P2	P2	Java Loops	Practice Java's loops
P3	P3	Drawing App	Practice writing Java static methods and nested loops
P4	P4	Passport Class	Practice basic class definition
-	P5	Photo Processing System	Practice writing classes, using 'this', exceptions, StringBuffer, and parsing

			strings
P5	P6	PhotoManager	Practice class definition, use of a class by another class, ArrayLists and Collections class
P6	P7	Array Utilities	Practice one-dimensional arrays
P7	P8	Diagram System	Practice two-dimensional arrays and interfaces
P8	P9	Media Rental Manager	Practice design

4.4.1 Design Process

The project slides were new material that was introduced in Fall of 2023 that previously had not existed. The creation of these slides were guided by the UDL principles equitable use, flexibility in use, simple and intuitive, perceptible information, and low physical effort. The simple and intuitive and low physical effort principle was applied to the project slides as the goal of the project slides was to give students a tool to help understand the project descriptions without showing them how to write the code. Low physical effort, flexibility in use, simple and intuitive, and equitable use was applied when deciding to use color to further help students understand the project descriptions. Simple and intuitive, equitable use, and perceptible information was applied by giving the students access to the project slides in PDF and an animation in MP4 format. See 6.1.1 for more discussion on how these principles guided the design process.

When designing the project slides, the first decision to be made was how to represent code. Due to the nature of the programming projects, the slides needed to be a tool that the students could use to help guide them on how to write the code, but could not be a tool that showed them exactly how to write Java code. Pseudocode could have been used, but pseudocode is a representation of code that is not language specific. Therefore if the students had access to pseudocode, they would not be learning the process of creating

code. Instead, they would be learning how to turn pseudocode into Java code which was not the objective of the programming projects. Therefore, simplistic English was chosen to guide students through the code that they would need to write.

To further help guide students on what the code was expected to do, color was used consistently to help students recognize parts of the code like variables, if statements, and method calls. Color was used to help students follow the examples. Throughout the slides, consistent colors were used so students would find it easier to understand more complex projects as the course continues. Table 4.2 shows the meaning of each color that was used. Whenever colors were used as input, like G for green, the appropriate color was matched to the word to help students quickly understand.

Due to colorblindness, it was very important to make sure that the colors were distinguishable from each other even if a student did not have the full color spectrum. The slides were shown to a 62 year old man that has red-green colorblindness. Red-green colorblindness is the most common form of color vision deficiency (*Types of Color Vision Deficiency | National Eye Institute*, n.d.). Each color was displayed and asked if they were distinguishable. In order to further check that important colors were distinguishable, a grayscale filter was applied to check that all colors would be readable.

Table 4.2 The colors of text used in the project slides and the meanings of that text

Color	**Meaning**
Blue	Emphasis Color
Red	Variable Names
Tan	If Statements and Loops
Purple	Method and Class Names
Grey	*null*

Pink	Comments

Images and objects were used whenever possible to help students visualize the code. For example, one of the projects required students to calculate the area of a triangle. Therefore, a triangle was used to show how the variables were associated as seen in Figure 4.17. 1 and 2 dimensional Arrays and databases were represented with tables as seen in Figure 4.18, Figure 4.19, and Figure 4.20, respectively.

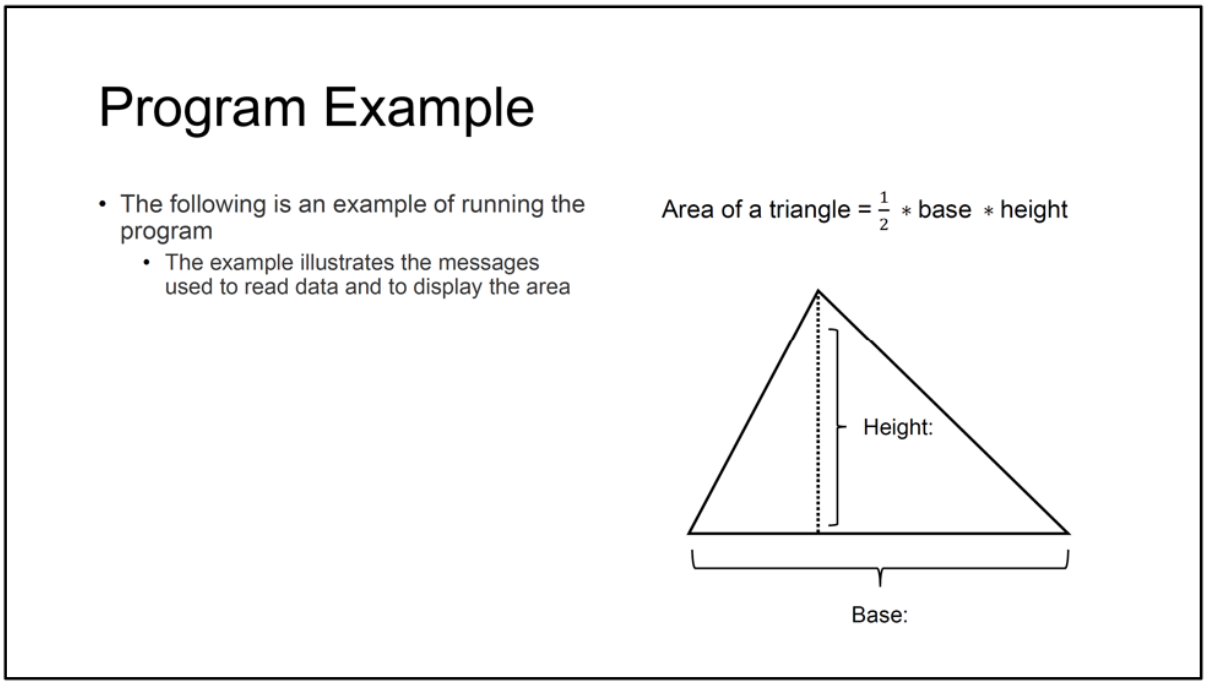


Figure 4.17
A slide showing images used when possible

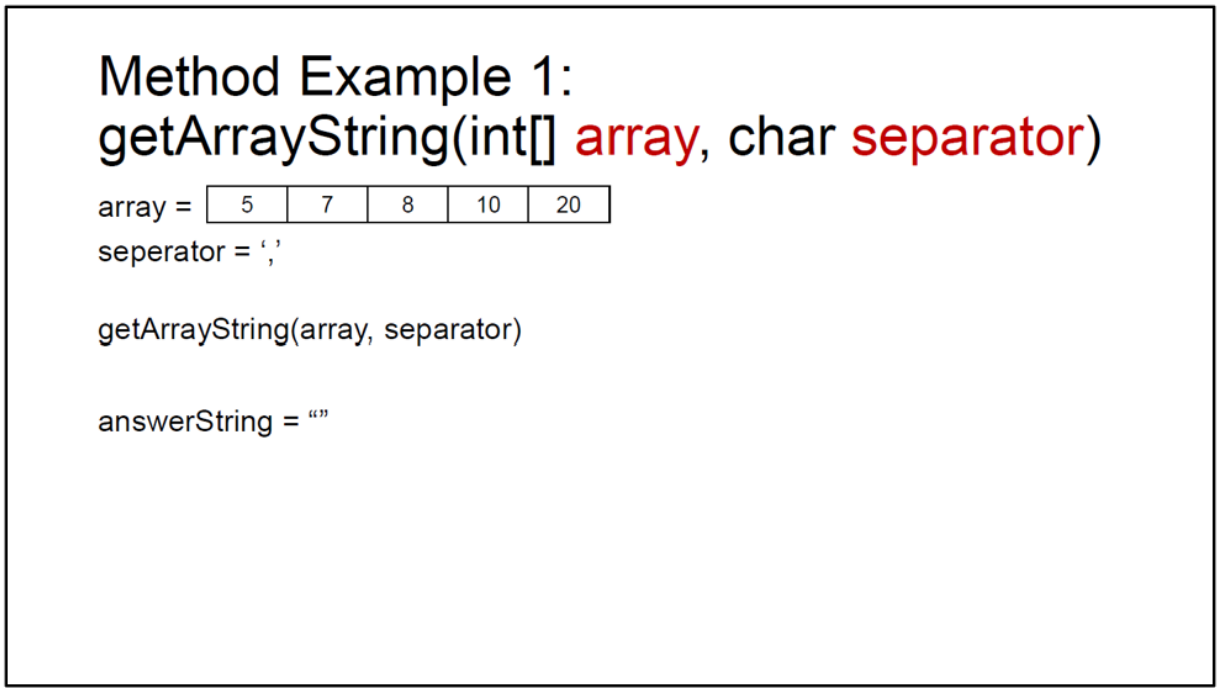


Figure 4.18
A slide showing how 1 dimensional arrays were represented

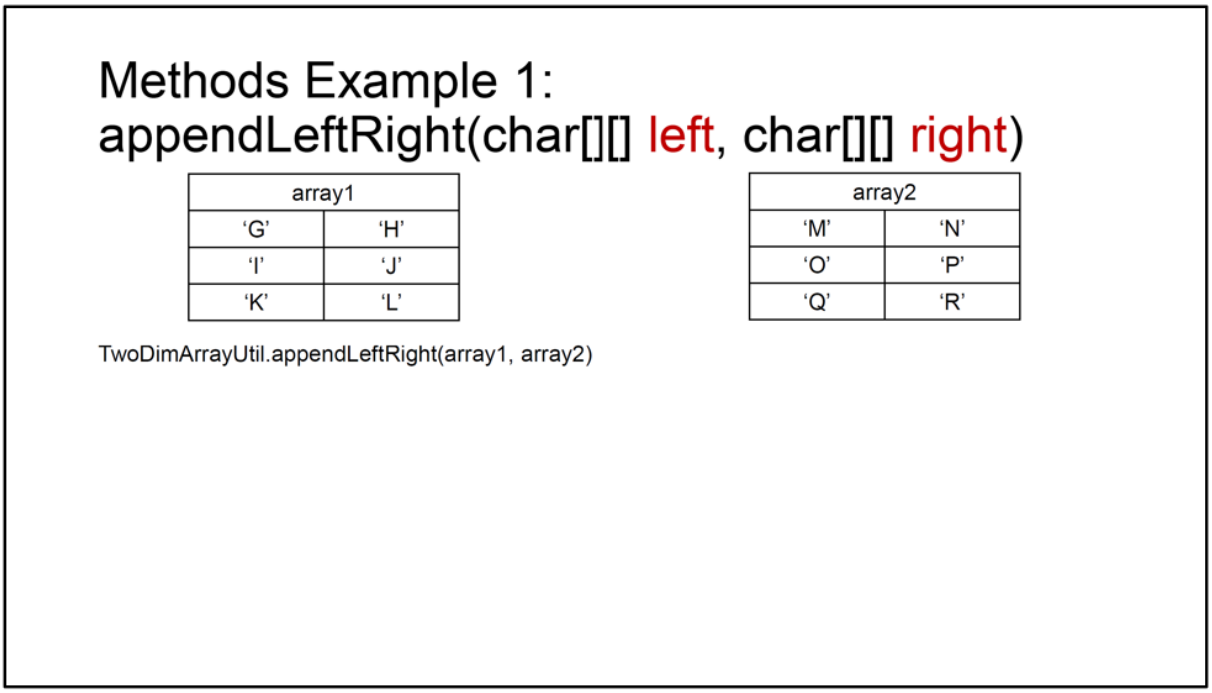


Figure 4.19
A slide showing how 2 dimensional arrays were represented

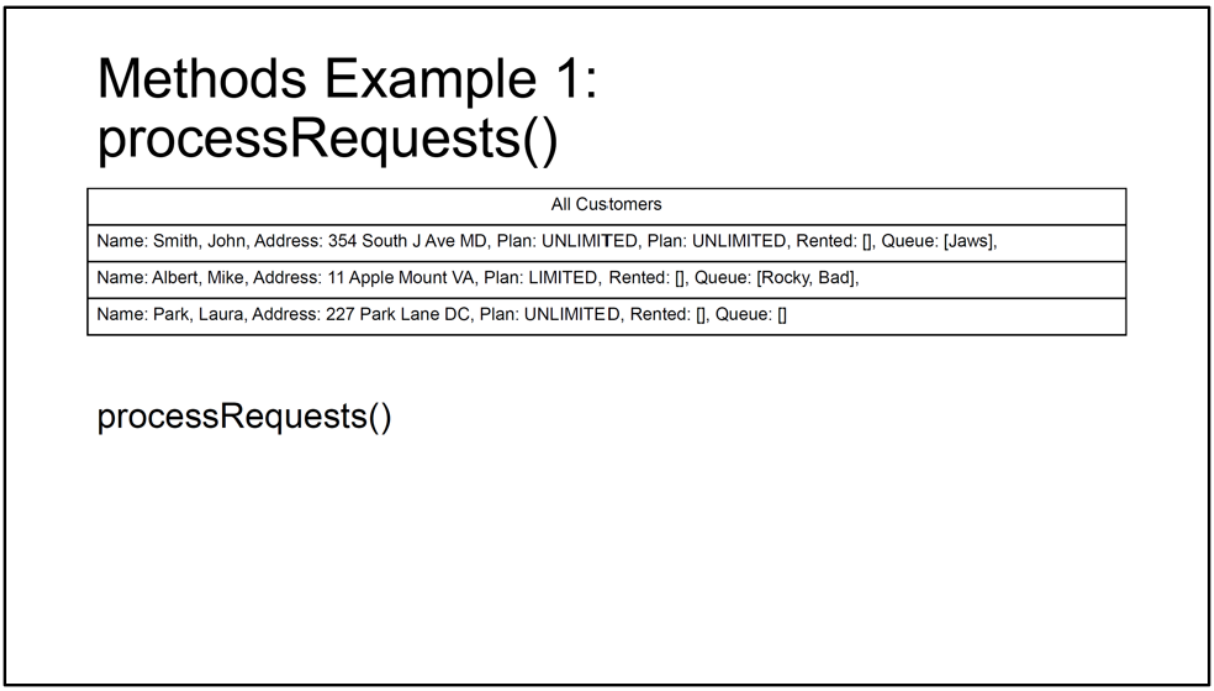


Figure 4.20
A slide showing how databases were represented

4.4.2 Specifications

The course website contained individual webpages for each project. There were two different ways that the students were given specifications for the project code. During the beginning of the course, the webpage for the project would give an explanation for each program, method, or constructor that the student was expected to implement. As the course continued, the specifications would exist in a Javadoc to explain the method or constructor expected for each class. These specifications would sometimes include examples. If the project required students to create a class where multiple methods depend on each other, a sample driver might be supplied instead of a single example.

Similar to the lecture slides, there are ways to improve the way the specifications were presented to the students to help with comprehension. As seen in Figure 4.21, the specifications that were given to students on the webpage were blocks of text with only small formatting like bolding words to help students understand important aspects of the specification of method.

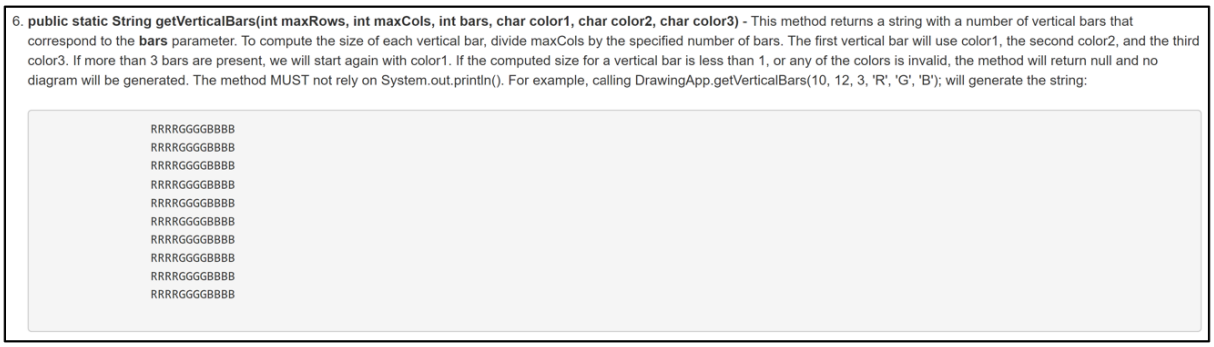


Figure 4.21
Specifications for a method on the course website

The formatting of the Javadoc does improve guidance of students through the explanation of each constructor and method. For example, Javadocs give a short summary for the constructor and methods expected. As seen in Figure 4.22, the summary gives a list of the constructor's names and their expected inputs as well as short descriptions of the constructors.

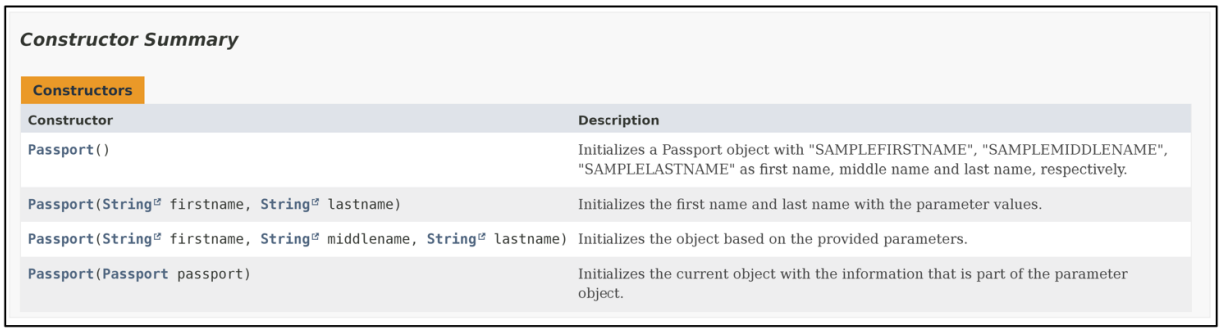
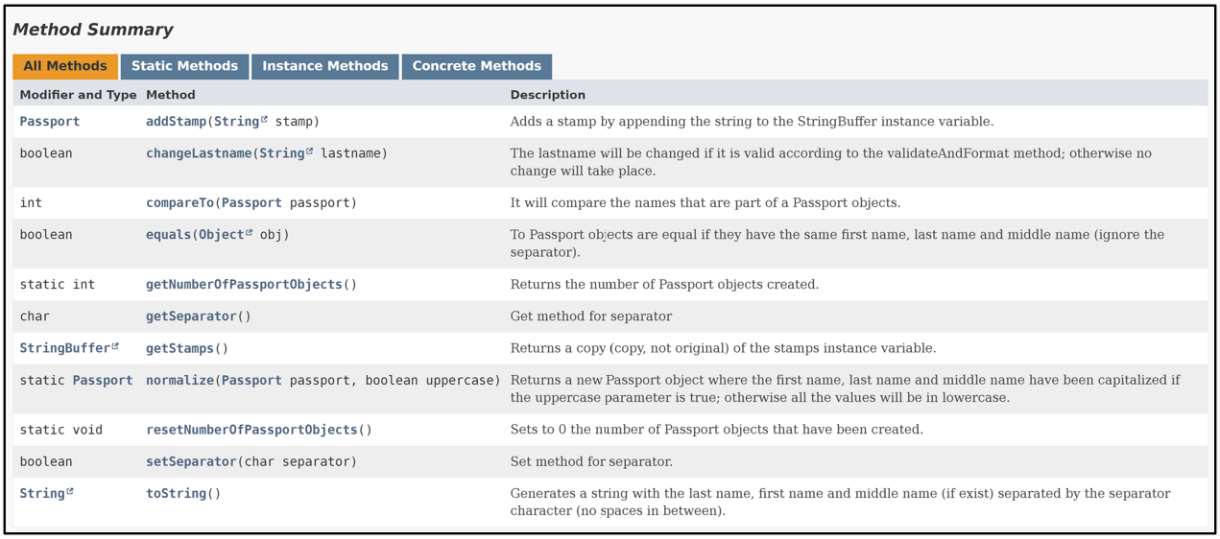


Figure 4.22
Javadoc Constructor Summary



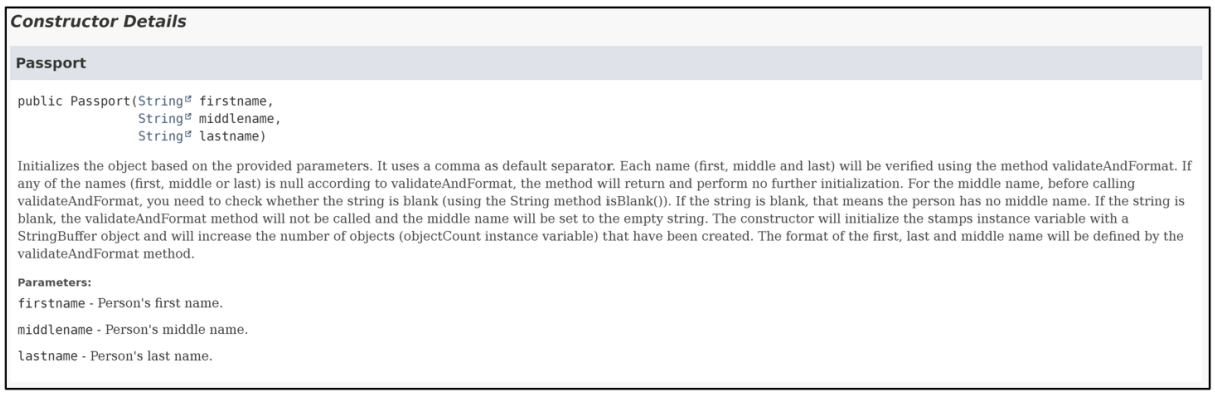
Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
Passport	addStamp(String stamp)	Adds a stamp by appending the string to the StringBuffer instance variable.
boolean	changeLastname(String lastname)	The lastname will be changed if it is valid according to the validateAndFormat method; otherwise no change will take place.
int	compareTo(Passport passport)	It will compare the names that are part of a Passport objects.
boolean	equals(Object obj)	To Passport objects are equal if they have the same first name, last name and middle name (ignore the separator).
static int	getNumberOfPassportObjects()	Returns the number of Passport objects created.
char	getSeparator()	Get method for separator
StringBuffer	getStamps()	Returns a copy (copy, not original) of the stamps instance variable.
static Passport	normalize(Passport passport, boolean uppercase)	Returns a new Passport object where the first name, last name and middle name have been capitalized if the uppercase parameter is true; otherwise all the values will be in lowercase.
static void	resetNumberOfPassportObjects()	Sets to 0 the number of Passport objects that have been created.
boolean	setSeparator(char separator)	Set method for separator.
String	toString()	Generates a string with the last name, first name and middle name (if exist) separated by the separator character (no spaces in between).

Figure 4.23
Javadoc Method Summary

Similarly, as seen in Figure 4.23, the Javadoc gives a summary of the modifier and type for the method, the method and its expected inputs, and a short description for each method. However, when students scroll down to access the constructor details or method details as seen in Figure 4.24, a similar problem appears with large blocks of text with limited formatting appearing.



Constructor Details

Passport

```
public Passport(String firstname,
                String middlename,
                String lastname)
```

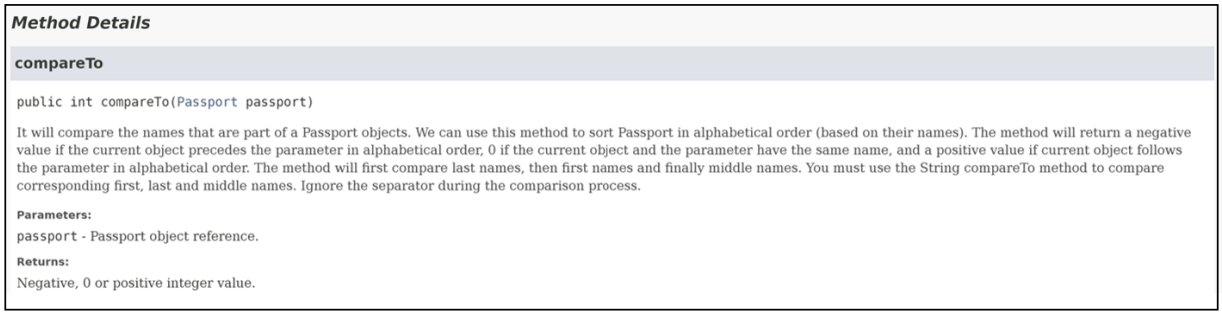
Initializes the object based on the provided parameters. It uses a comma as default separator. Each name (first, middle and last) will be verified using the method validateAndFormat. If any of the names (first, middle or last) is null according to validateAndFormat, the method will return and perform no further initialization. For the middle name, before calling validateAndFormat, you need to check whether the string is blank (using the String method isBlank()). If the string is blank, that means the person has no middle name. If the string is blank, the validateAndFormat method will not be called and the middle name will be set to the empty string. The constructor will initialize the stamps instance variable with a StringBuffer object and will increase the number of objects (objectCount instance variable) that have been created. The format of the first, last and middle name will be defined by the validateAndFormat method.

Parameters:

`firstname` - Person's first name.

`middlename` - Person's middle name.

`lastname` - Person's last name.



Method Details

compareTo

```
public int compareTo(Passport passport)
```

It will compare the names that are part of a Passport objects. We can use this method to sort Passport in alphabetical order (based on their names). The method will return a negative value if the current object precedes the parameter in alphabetical order, 0 if the current object and the parameter have the same name, and a positive value if current object follows the parameter in alphabetical order. The method will first compare last names, then first names and finally middle names. You must use the String compareTo method to compare corresponding first, last and middle names. Ignore the separator during the comparison process.

Parameters:

`passport` - Passport object reference.

Returns:

Negative, 0 or positive integer value.

Figure 4.24
Javadoc Constructor and Method Details

It is important for materials that are given to students to be consistent. Rather than changing the wording of the specifications, the formatting was changed to highlight important parts of the specification using the color scheme mentioned above. While the text is the same as the text on the website or Javadoc, the format of how that text was presented to students was different on the project slides. Figure 4.24 shows the original format of the Javadoc. Figure 4.25 shows the reformatted identical text that appeared in the project slide packet. Instead of showing the students the information in one large paragraph, the information was broken up into multiple bullet points so students could focus on each sentence without getting overwhelmed as seen in Figure 4.25.

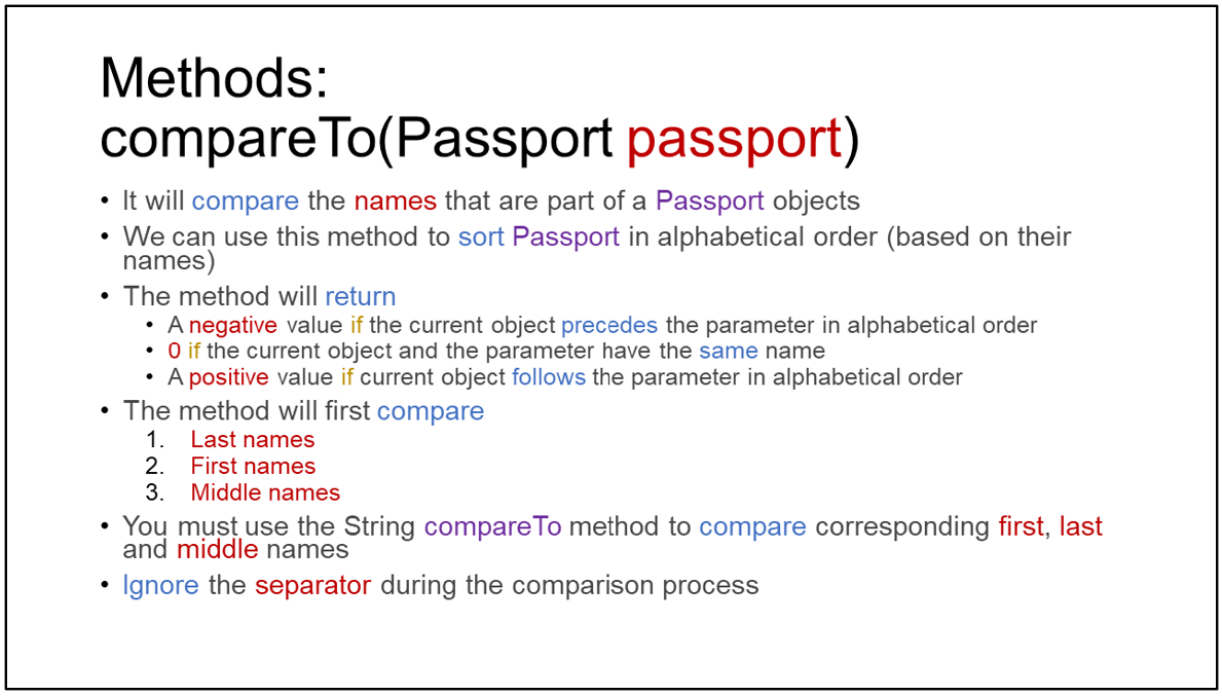


Figure 4.25
A slide showing how specifications were represented

4.4.3 Examples

Previously, the examples that were given to the students were very scarce. When there was an example for a specific method, often it was only a single example as seen in Figure 4.21. Otherwise, examples were often given in a sample driver with output as seen in Figure 4.26. A sample driver was a code snippet that students could run to check that their code was running as expected. While this is helpful to check code, it often did not give any information as to what is incorrect if the code was not working as expected or in edge cases that should throw exceptions. The release and public tests were also useful for showing the student if their code was performing as expected, they also would not give information when the code was incorrect.

Driver

```
public class Driver {

	public static void main(String[] args) {
		Passport.resetNumberOfPassportObjects();

		Passport passport1 = new Passport("claudia", "I.", "smith");
		System.out.println(passport1);

		Passport passport2 = new Passport("John", "RoberTs");
		System.out.println(passport2);

		Passport passport3 = new Passport();
		System.out.println(passport3);
		System.out.println("=============");

		passport1.addStamp("Spain");
		passport1.addStamp("London");
		System.out.println("Stamps for " + passport1 + "-->" + passport1.getStamps());

		passport1.setSeparator('#');
		System.out.println(passport1);
		Passport passport4 = new Passport("CLAUDIA", "I.", "smith");
		System.out.println(passport1 + " same to " + passport4 + "? " + passport1.equals(passport4));

		System.out.println("=============");
		System.out.println("Comparing");
		System.out.println("Comp1: " + (passport1.compareTo(passport2) > 0));
		System.out.println("Comp2: " + (passport2.compareTo(passport1) < 0));
		System.out.println("Comp3: " + (passport1.compareTo(passport4) == 0));

		System.out.println("=============");
		System.out.println("Normalizing");
		Passport normalized1 = Passport.normalize(passport1, true);
		System.out.println("Normalize #1: " + normalized1);
		System.out.println("Normalize #2: " + Passport.normalize(passport1, false));
		System.out.println("Normalize #3: " + Passport.normalize(passport4, false));

		System.out.println("Number of objects: " + Passport.getNumberOfPassportObjects());
	}
}
```

```
Output
Smith,Claudia,I.
Roberts,John
Samplelastname,Samplefirstname,Samplemiddlename
=============
Stamps for Smith,Claudia,I.-->SpainLondon
Smith#Claudia#I.
Smith#Claudia#I. same to Smith,Claudia,I.? true
=============
Comparing
Comp1: true
Comp2: true
Comp3: true
=============
Normalizing
Normalize #1: SMITH#CLAUDIA#I.
Normalize #2: smith#claudia#i.
Normalize #3: smith,claudia,i.
Number of objects: 7
```

Figure 4.26
Example Driver with Output on Course Website

To help students comprehend what the project is asking of them further, multiple examples were added to the project slides that originally were not on the website or Javadoc for each method and constructor when needed. The first example given to students followed any examples that the professor had given them in order to keep with consistency as seen in Figure 4.27. The next examples would follow to show students what would happen if incorrect inputs were passed to the method, exceptions were thrown, an if statement was not entered, a loop was not entered, ect. as seen in Figure 4.28.

getVerticalBars Example 1

- DrawingApp.getVerticalBars (10, 12, 3, 'R', 'G', 'B'); will return the following string:

maxRows = 10
maxCols = 12
bars = 3
color1 = 'R'
color2 = 'G'
color3 = 'B'

```
public static String getVerticalBars(10, 12, 3, 'R', 'G', 'B')
    diagram = "RRRRGGGGBBBB
               RRRRGGGGBBBB
               RRRRGGGGBBBB
               RRRRGGGGBBBB
               RRRRGGGGBBBB
               RRRRGGGGBBBB
               RRRRGGGGBBBB
               RRRRGGGGBBBB
               RRRRGGGGBBBB
               RRRRGGGGBBBB"
    return diagram
```

Figure 4.27
A slide showing the example from course webpage

getVerticalBars Example 4

- If the computed size for a vertical bar is less than 1, or any of the colors is invalid, the method will return null and no diagram will be generated

maxRows = 5
maxCols = 3
bars = 4
color1 = 'R'
color2 = 'G'
color3 = 'B'

```
public static String getVerticalBars(5, 3, 4, 'R', 'G', 'B')
    Are color1, color2, color3 valid colors? Yes
    We need to compute the size of each vertical bar:
        eachBarSize = 0
    Is eachBarSize < 1? Yes
    return null
```

Figure 4.28
A slide showing an added example showing other possible inputs that would lead to no diagram returned

4.4.4 Output

Once the project slides were created, they were distributed to the students in two formats on the course website. The first format was an MP4 file with no audio. This was provided so that students could watch the slides and how the program is supposed to run in an animation format. The purpose was to allow students to be able to understand the chronological process that the code would complete that is not often visible to the programmer. Each slide was shown to the students for 5 seconds. This speed was chosen so that slower readers would be able to read all text on the screen before moving on to the next screen. This speed also allowed students with slower processing speeds to comprehend the decisions the code meant to be making at any given time. Having the animation in an MP4 format also allowed students more control if needed since they would be able to speed up or slow down the animation, pause, fastforward, or rewind the animation as needed. Many other animation formats do not have these features, so students would have to wait for the animation to loop back to the beginning to get back to the frames they would like to study closer. Therefore, the video format was chosen to give students as much control as possible.

The second format available to students was a pdf of the project slides. Since the lecture slides were also provided to the students in pdf format, the projects were exported in pdf format to be consistent. While the animation output provides context for what the code is meant to do when run, the pdf was meant to mimic stepping through the code similar to Eclipse's debug tool. This allows students to stop in between code executions to fully understand what work was done at a specific point in the code. The animation video also has the disadvantage of the students not being able to use screen readers for text on

the screen. Providing a pdf version allows students to use accessibility tools like screen readers as needed.

UDL principles were implemented into the course's materials by modifying the course's lecture slides and designing project slides. The lecture slides were updated using built-in accessibility tools like Microsoft Powerpoint's Accessibility Checker, Spelling and Grammar Checker, and the Flesch Reading Ease test. Manual modifications guided by the web content accessibility guidelines and recommendations from dyslexic associations were made. These changes included adding white space to all slides, changing the format of the lists, and changing the color and font of text to improve readability. The project slides were designed by using simplistic English to explain what the code should do, text color to highlight important parts of the code, and images and objects whenever possible to show a visual representation of the code. The text from the original project descriptions was made more readable by using white space. More examples of methods and constructors were created. The project slides were provided to students in two different formats, PDF and MP4, so that students could choose the format that helps them the most.

Chapter 5: Survey and Interview Findings

This chapter discusses what findings were found from the data collected from students and TAs. The first section presents an analysis of data collected to answer the second research question "How does updating an introductory programming course to follow the universal design for learning framework affect the experience of students?". The data collected for this section came from surveys, grades, and interviews.

To answer the second and third research question, Fall 2022 and Fall 2023 CMSC 131 students were compared. To make sure the two student populations were not different along any meaningful demographic dimensions, responses were compared between the two populations using a Mann–Whitney U test as all data was categorical. Five Mann–Whitney U tests were applied to each question. The different groups that were compared can be seen in Table 3.12.

After completing the analysis, there were no statistically significant differences for any of the questions. Therefore, the demographics for the two student populations for all five tests could not be confirmed to be different in whether the students were first generation college students, had English as their first language, worked during the semester and how many hours on average, have previous experience with programming and how many computer science courses the students took, knew of an adult working in a computer science related field when growing up, were in gifted or honors programs, were in special education programs, had a private tutor, and were registered with disability services at UMD.

Table 5.1 Number of students that completed the demographics survey vs the number of students participating in the study

	Fall 2022 CMSC 131	**Spring 2023 CMSC 132**	**Fall 2023 CMSC 131**

Number of Students that took the Demographics Survey	109	17	19
Total Number of Students Participating in Study	149	33	63

The second section discusses the experiences of neurodivergent and neurotypical students during all three semesters and answers the final research question "What are the experiences of neurodivergent and neurotypical students after updating an introductory programming course to follow the universal design for learning framework?". The data collected for this section came from surveys and interviews.

Along with looking at the outcomes of the UDL modifications, additional student needs were found. In the third section, the needs of the students beyond the UDL modifications includes wanting group projects, more examples and practice projects, solutions to exams and projects with adequate explanations, study guides, more help from teaching staff, slowing the pace of the course, and better placement for students' abilities to be assigned to correct courses. Interviews from Fall of 2022 and Fall of 2023 were also analyzed during this section to get a better understanding of the survey responses about what the students need.

5.1 How updating CMSC 131 Affects Student Experiences

This section analyzes data to answer the question: *how does updating an introductory programming course to follow the universal design for learning framework affect the experience of students?* Survey answers and grades from Fall of 2022 and Fall of 2023 were analyzed. Overall, the students in Fall of 2023 had a better experience than the students in Fall of 2022. The survey data showed that the Fall of 2023 students felt they

understood topics more than Fall of 2022, the Fall of 2023 students felt the quizzes more accurately showed how much they understood the material than the Fall of 2022 students, the Fall of 2023 students found the slides more helpful than the Fall of 2022 students, the Fall of 2022 found the slides less helpful than Fall of 2023 students. The students in Fall of 2023 received better grades than the students in Fall of 2022 in four projects, a debugging quiz, and three Exams. This led the Fall 2023 students to earn higher final grade averages than the Fall of 2022 students. The following sections present the data and analysis that led to these conclusions.

5.1.1 Survey Findings

Based on the Quantitative Survey Data, the students in Fall of 2023 reported having better experiences in CMSC 131 than the students in Fall of 2022. In Fall of 2023, the students felt they understood two topics, Java Memory Maps and Java's Interfaces, better than the students in Fall of 2022, as seen in Table 5.2. The perception of knowing Java Memory Maps was significantly higher in Fall of 2023 compared to Fall 2022, $z = 2.01$, $p = 0.044$. The median score on a likert scale from 1 to 5 was 3.34 in Fall of 2022 compared with 3.86 in Fall of 2023. The perception of knowing Java's Interfaces was significantly higher in Fall of 2023 compared to Fall 2022, $z = 2.05$, $p = 0.040$. The median score on a likert scale from 1 to 5 was 3.67 in Fall of 2022 compared with 4.2 in Fall of 2023.

In Fall of 2023, students also reported higher levels of satisfaction with the quizzes, responding that they felt that the quizzes accurately show how much they understand the material better than the students in Fall of 2022. The perception of quizzes was significantly higher in Fall of 2023 compared to Fall 2022, $z = 2.85$, $p = 0.0044$. The median score on a likert scale from 1 to 5 was 3.68 in Fall of 2022 compared with 4.25 in Fall of 2023.

Table 5.2 Fall of 2023 students had a better experience with 2 topics and the quizzes

Question	**Fall 2022 CMSC 131 Average**	**Fall 2023 CMSC 131 Average**	**P - Value**	**Z-Score**
I understand the topic Java memory map.	3.34	3.86	0.044	2.01
I understand the topic Java interfaces.	3.67	4.2	0.040	2.05
I feel that the quizzes accurately show how much I understand the material.	3.68	4.25	0.0044	2.85

Based on the Qualitative Survey Data, the students in Fall of 2023 reported having better experiences with the updated material than the students in Fall of 2022. When asked *What type of course material do you find is the most helpful? Why?*, 14% of the responses in Fall of 2022 said the lecture slides were helpful. In Fall of 2023, 20.34% of the responses in Fall of 2023 said the lecture slides were helpful. The increase from 14% in Fall of 2022 to 20.34% in Fall of 2023 suggests the modifications to the lecture slides improved the usability of the slides.

The students said the slides were helpful in Fall of 2022 because the information on the slides was clear and the slides were easy to reference, as seen in table 5.3. In Fall of 2022, 38.46% of the responses stated the information was clear. The slides being clear helped to refresh the students' memory about material. The students reference the slides to help them with writing code. A student stated, "I find that the slides are the most helpful because they clearly list out the rules for every type of function". Often students would use the sample code and slides together to help them understand code. A student stated "Going back between the sample code and the slides to compare, contrast, and understand things better."

23.08% of the responses said the slides were easy to reference. The students used the slides to find specific information quickly. A student stated, "I think the slides are the most helpful because they are easier to reference when you are trying to find something specific." Students often referred back to the slides while writing their code. A student stated, "Slides and example code are really helpful because I can refer back to the content and see how it works."

The students said the slides were helpful in Fall of 2023 because the information on the slides were clear, the slides were easy to reference, and the slides were easy to read, as seen in Table 5.3. In Fall of 2023, 25% of the responses stated the information was clear. A student stated, "I like the slides the most because they are very direct in what information is necessary to know." The clarity of the slides helped students to know what material they should focus on. Students also mentioned that the slides were clear without being too simplistic. A student stated, "The slides as it contain the information in an organized manner. Not too complicated with a lot of information and not too simple with minimal information."

25% of the responses said the slides were easy to reference. The students liked that the lecture slides were viewable outside of class because it makes necessary information easy to find. A student stated, "The slides; needed information is easily accessible." The students used the lecture slides to review content. One student stated, "The slides are very helpful. They are easy to review". Another student agreed with the first while also mentioning that using the slides and sample code together helped them to review content. They stated, "The examples and the slides because I can review both outside of lecture."

33.33% of the responses said the slides were easy to read. The slides being easy to read is important as students often need a way to find information quickly when completing assignments or studying. A student stated, "they are quick to the point and easy to

read/comprehend". Another student agreed and stated, "I like how the slides consolidate relevant information into one setting". The students' responses to the slides in Fall of 2023 suggest that the modifications to the lecture slides improved the students' experiences compared to the students in Fall of 2022.

Table 5.3 Percentage of student responses for each code from the question What type of course material do you find is the most helpful? Why?

What type of course material do you find is the most helpful? Why?	**Fall 2022 CMSC 131 Percentage**	**Fall 2023 CMSC131 Percentage**
Information is Clear	38.46%	25%
Easy to Reference	23.08%	25%
Easy to Read	0%	33.33%
Total Amount of Responses about Slides	14	12

When asked *What type of course material do you find is the least helpful? Why?*, 33.00% of the responses in Fall of 2022 said the lecture slides were the least helpful. In Fall of 2023, 28.81% of the responses in Fall of 2023 said the lecture slides were the least helpful. This decrease suggests that the modifications of the lecture slides improved the usability of the slides. The students said the slides were least helpful in Fall of 2022 and Fall of 2023 because there were not enough examples on the slides, the students didn't use them after lecture, and the slides are hard to follow, as seen in Table 5.4.

In Fall of 2022, 36.37% of the responses stated there were not enough examples on the slides. Students want more examples in the lecture slides to show them how to code. A student stated, "I find the slides to be the least helpful as they usually do not have a ton of information on them, they cant really show you how to code." Students found that the slides are often only useful as they are only helpful for conceptual information. A student stated, "Slides I guess? It's not unhelpful but really only the best for conceptual stuff." Another

student agreed and stated, “powerpoints because the content on there does not put the actual code in action but it is still helpful”. The slides only containing conceptual information makes it difficult for students to retain information. A student stated, “Slides since it is hard to just remember the information without practicing it”

33.33% responses stated the students didn’t use them after lecture. Students found that the slides were not helpful outside of lecture. A student stated, “The power points don't do much for me, and I never look at them outside of class, because usually there is sample code or the following examples that go into more depth on the concept rather than just briefly explaining it.” Another student agreed and stated that the lecture slides are the least helpful as they only complement the lecture that day. They stated, “Slides are the least helpful. They do help me a little bit, but they really are supplementary pieces of information alongside whatever lecture is being discussed that day.” Students also found the slides to not be useful because they were boring. A student stated, “The slides, I just get bored of slides easily”.

24.24% of the Fall 2022 responses said the slides were hard to follow. The students found that the slides were disorganized. A student stated, “the slides. they are difficult to sort through and organize when reviewing.” Another student agreed and stated, “The slides but only because they can be hard to understand on their own, they are helpful when gone over by [the instructor] and in combination with the sample code.” Students also found the slides hard to read. A student stated, “Slides, because often they are hard to read and relatively vague.”

In Fall of 2023, 52.17% of the responses stated there were not enough examples on the slides. The students wanted the slides to have more examples to help them understand code better. A student stated, “The powerpoints are the least helpful. It would be better if there were some coding examples with all the slides so you can see how things are

implemented.” Another student agreed and stated, “I like the slides, but they are least essential to my learning since convey raw expository information that is difficult to interpret in the absence of a hard example”. Students felt the lack of examples made the slides very abstract. A student stated, “Slides because it is more general information more than how to actually apply it.”

21.74% of the responses stated the students didn’t use them after lecture. One reason that the students did not use the slides after lecture was because they do not learn from slides. One student stated, “I often do not look back at slides because just reading the text on the slides is disengaging for me”. Another student agreed and stated, “I don't use the slides after class, since parsing text is not as helpful for me, but they are useful during lecture as they accompany spoken explanation.” Even though students stated that they did not use them after lecture, students still had a positive experience with them. A student stated, “Honestly all of them are super helpful! But since I listen to lectures in class I don't go to the slides that often. I mostly practice the codes and visit the class webpage a lot. But that doesn't mean the slides are not helpful!”. Another student agreed and stated, “Slides because though they are still helpful not very helpful. ” Some students did not have a particular reason for not looking back on the slides. One student stated, “If anything slides and it's nothing in particular I just don't really look back on them.”

4.34% of the Fall of 2023 responses said the slides were hard to follow. Some students found that the slides were not useful without the accompanying lecture. One student stated, “In isolation the slides are a bit hard to follow sometimes because all the information is not written on them so instruction is necessary to get the most out of them.”

Table 5.4 Percentage of student responses for each code from the question What type of course material do you find is the least helpful? Why?

What type of course material do you find is the	**Fall 2022**	**Fall 2023**

least helpful? Why?	**CMSC 131 Percentage**	**CMSC 131 Percentage**
Not enough examples	36.37%	52.17%
Don’t use them after lecture	33.33%	21.74%
Hard to Follow	24.24%	4.35%
Total Amount of Responses about Slides	33	17

In Fall of 2022 and Fall of 2023, the students were asked in each of the after-exam surveys, the short answer question: *Do you feel your programming assignment code reflects your understanding of the material?* This question connects to the project slides as the project slides were meant to make the programming assignments easier. So, if students in Fall of 2023 felt their code reflected their understanding more of the material that could suggest that the project slides helped students to have better experiences with the programming projects.

In Fall of 2022, 77.92% of responses stated that their programming assignment code does reflect their understanding, 19.48% of responses stated that their programming assignment code somewhat reflects their understanding, and 2.60% of responses stated that their programming assignment code does not reflect their understanding.

In Fall of 2023, 80.36% of responses stated that their programming assignment code does reflect their understanding, 14.29% of responses stated that their programming assignment code somewhat reflects their understanding, and 5.36% of responses stated that their programming assignment code does not reflect their understanding. Therefore, the students in Fall of 2023 seem to have felt their programming assignment code reflects their understanding of the material about the same compared to Fall of 2022.

Table 5.5 Percentage of student responses for the question Do you feel your programming assignment code reflects your understanding of the material?

Do you feel your programming assignment code reflects your understanding of the material?	**Fall 2022 CMSC 131**	**Fall 2023 CMSC 131**
Yes	77.92%	80.36%
Somewhat	19.48%	14.29%
No	2.60%	5.36%
Total Amount of Responses	77	56

When asked *Do you feel your programming assignment code reflects your understanding of the material? Why?*, Responses in the Fall 2023 of the study were the same as Responses in Fall of 2022, as seen in Table 5.6. In Fall of 2022, the responses that were positive mentioned that the programming assignments implements topics from lecture, improves understanding, shows the students own ability, had lots of time for completion, were a fair difficulty, and helped to learn what the code is doing. In Fall of 2023, the responses that were positive mentioned that the programming assignments implements topics from lecture, improves understanding, shows the students own ability, had lots of time for completion, helped to learn what the code is doing, and simulates coding scenarios in the future.

When looking at the students' perceptions of how well the assignments reflected their programming knowledge, 10% of the responses in Fall of 2022 mentioned the programming assignments showed their knowledge and ability compared to 29.49% of the responses in Fall of 2023. This suggests the project slides helped students to show their ability to code.

Fall of 2022 students felt that their code reflects their logical thinking. They stated, "Yes because it shows our own unique thinking process." Another student agreed and stated, "Yes because it is a reflection of problem solving over time, which is what a practical programmer does". Therefore, students felt like the programming assignments helped to

simulate real programming. A student stated, "yes, much better than the exams do because they reflect practical use of code". A student did mention frustration with not being able to fully show their logical thinking. A student stated, "Generally yes as it allows me to come up with and then optimize my own solution to a problem. Often for this specific class I felt that the requirements to pass some tests restricted me to certain options when implementing. The last project felt the most free but I still found myself changing code to pass tests compared to my original solution, and this resulted in the feeling that my code was messy because my solution had to be converted to fit the test." Students often did not work with others on the programming assignments, so it shows their own ability. A student stated, "Yes because I try to work on them alone without consulting my friends".

In Fall of 2022, only 2.5% of the responses mentioned they had lots of time to complete the programming assignments while in Fall of 2023, that number increased to 8.97% of the responses mentioning they had lots of time to complete the programming assignments. Therefore, the project slides may have decreased the time needed to complete the projects. Fall 2022 students often discussed that they did not have enough time to complete the projects even if they understood the assignment. A student stated, "I do not believe it reflects my understanding. This is because sometimes I do not have enough time to finish projects as I have to juggle school, work, and commuting."

Fall of 2023 students often mentioned that they had enough time to complete the projects. A student stated, "Yes, because I have enough time to complete the code, and am able to do it to the best of my ability." Another student agreed and stated, "yes, plenty of time to work and demonstrate my knowledge." Students also mentioned that they had enough time to also go to office hours if they needed help with their project code. A student stated, "I do think they did. I had enough time to put my full effort into them and if I hit any roadblocks I was able to attend office hours and sort it out."

Table 5.6 Percentage of student responses for each code from the question Do you feel your programming assignment code reflects your understanding of the material? Why? (Positive Responses)

Do you feel your programming assignment code reflects your understanding of the material? Why? Positive Responses	**Fall 2022 CMSC 131 Percentage**	**Fall 2023 CMSC 131 Percentage**
Implements topics from lecture	38.75%	30.77%
Improves Understanding	26.25%	10.26%
Shows the Students own Ability	10%	29.49%
Had Lots of Time for Completion	2.5%	8.97%
Were a Fair Difficulty	1.25%	0%
Helped to Learn What the Code is Doing	2.5%	1.28%
Simulates Coding Scenarios in the Future	0%	1.282%
Total Amount of Responses	65	64

In Fall of 2022, the responses that were negative mentioned that the students were confused about code, the programming assignments were too difficult and had a bad structure, there were inconsistencies with what is being asked of the students between the exams and programming projects, the student felt like they were missing something, and there was not enough time to complete the projects. In Fall of 2023, the responses that were negative mentioned that the students were confused about code, the programming assignments were too difficult and had a bad structure, the programming assignments did not implement topics from the lecture, the student felt like they were missing something, and the programming assignments were stressful.

In Fall of 2022, 7.5% of the responses mentioned they were confused about the code they were writing compared to 1.3% of the responses in Fall of 2023. In Fall of 2022 students mentioned being confused about how to write the logic for the code even if they knew what to do. A student stated, "Most of the time, sometimes I understand the ideas

behind it but my grammar makes me fail a release test or something.” Other students were confused about what the requirements of the project was asking of them. A student stated, “Somewhat, it takes a little more time than I'd like to admit to fully understand what I need to implement and learn.”

In Fall of 2023, students that had no previous programming experience expressed that implementing the projects was confusing. A student stated, “For a beginner, everything is still very confusing and I have never done coding in my life so the programming assignments makes things more confusing instead of helping me learn step by step.” In another response, the statement stated, “I try to follow the examples from class and all to do the programming assignments but I am still confusing about everything.”

In Fall of 2022, 7.5% of the responses mentioned the programming assignments were too difficult and had a bad structure compared to 6.41% in Fall of 2023. In Fall 2022, students felt like the difficulty of the projects increased too quickly. “Kind of but sometimes I feel like the programming assignments are too difficult and each programming assignment has a big difficulty leap to the next one so I feel like it isn't steadily getting harder and harder but it's just much harder at once.”

In Fall 2023, Students thought that the lecture material did not match the difficulty of the projects. A student stated, “Mostly. I feel like I usually understand the lecture material, but the projects sometimes seem harder than what we learned in lecture”. A different student disagreed and stated, “I feel that students may benefit from more complex (both execution and error-handling-wise).”

In Fall of 2022, 1.2% of the responses mentioned they felt like they were missing something compared to 3.85% in Fall of 2023. In Fall of 2022, students felt their conceptual knowledge did not translate to their code. A student stated, “Not exactly, I feel like I'm

missing something in terms of translating understanding into execution." Another student also had a similar experience when using the class examples to help with the projects. They stated, "Sometimes. I usually use class examples to help me form my code but sometimes I don't understand why I am doing what I am doing." A different student agreed, but used the in class examples to experiment with the code and figure out what they were missing. They stated, "Generally yes, however there are some cases where I feel a little lost and have to experiment with the code. I do however think that this is probably the best way to cement the material, because in a real world programming situation you will have the programming software and its tools to tell you where you might have made a mistake, and you will also have the java or other language documentation to tell you how to use different methods etc..." A different student felt that because the projects were too similar to the in class examples which meant their code didn't reflect how much they knew. They stated, "Not really, because often the programming assignments code is very similar to the examples, which isn't a bad thing since I feel like there isn't much time to figure it out."

In Fall of 2023, one student mentioned that they were missing something because they were new to coding and didn't understand the material. They stated, "I don't understand anything still because I am new to coding." A different student mentioned that they were missing something because they didn't have time to understand the full concept. They stated, "No, the programming assignment code does not necessarily reflect understanding of material. It definitely helps me to learn the material much better. However, sometimes it causes more stress on students to finish the project than to understand the full extent of the concept being taught by the project."

Table 5.7 Percentage of student responses for each code from the question Do you feel your programming assignment code reflects your understanding of the material? Why? (Negative Responses)

Do you feel your programming assignment code	**Fall 2022**	**Fall 2023**

reflects your understanding of the material? Why? Negative Responses	**CMSC 131 Percentage**	**CMSC 131 Percentage**
Students were Confused about Code	7.5%	1.28%
Too Difficult and Had a Bad Structure	7.5%	6.41%
Inconsistencies Between the Exams and Programming Projects	1.25%	0%
Felt like They were Missing Something	1.25%	3.85%
Not Enough Time to Complete the Projects	1.25%	0%
Do Not Implement Topics from Lecture	0%	5.13%
Stressful	0%	1.28%
Total Amount of Responses	15	14

In Fall of 2022 and Fall of 2023, the students were asked in the after-exam survey, the short answer question: *Do you feel prepared for the exams?* In Fall of 2022, 60.98% of responses stated that they were prepared, 23.17% of responses stated that they were sometimes prepared, and 15.85% of responses stated they were not prepared. In Fall of 2023, 60.98% of responses stated that they were prepared, 23.17% of responses stated that they were sometimes prepared, and 15.85% of responses stated they were not prepared. Therefore, the students seem to have felt more prepared for the exams in Fall of 2023 compared to Fall of 2022. Therefore, the students in Fall of 2023, felt more prepared for the exams than the students in Fall of 2022.

Table 5.8 Response Percentages for the question Do you feel prepared for the exams

Do you feel prepared for the exams?	**Fall 2022 CMSC 131**	**Fall 2023 CMSC 131**
Yes	60.98%	75.44%
Sometimes	23.17%	15.80%
No	15.85%	8.77%

Total Amount of Responses	82	57

5.1.2 Grades Findings

The students in Fall of 2023 did significantly better than the students in Fall of 2022 in four projects, a debugging quiz, and three Exams. This led to the students in Fall of 2023 having higher final grades compared to the students in Fall of 2022 The students in Fall of 2022 did not do statistically significantly better on any of the projects, quizzes, or exams than the students in Fall of 2023.

In Fall of 2023, the students did significantly better than the students in Fall of 2022. The final grades from Fall 2022 (M = 84.13, SD = 14.55) and Fall 2023 (M = 88.29, SD =10.60) indicate the students in Fall 2023 did significantly better than the than the students in Fall of 2022, t(2) = 2.04, p = 0.043 as seen in Table 5.9.

Table 5.9 Comparison of Final Grades between Fall of 2022 and Fall of 2023

	Fall 2022 CMSC 131 Average/ Standard Deviation		**Fall 2023 CMSC131 Average/ Standard Deviation**		**T-statistic**	**P-Value**	**Cohen's d**
Final Grade	84.13	14.55	88.29	10.60	2.04	0.043	0.33

In Fall of 2023, the students did significantly better than the students in Fall of 2022 on four projects, as seen in Table 5.10. These projects were Java Loops, Drawing App, Passport Class (a), Photo Manager. There was a significant difference in the grades received in Fall of 2022 (M = 95.66, SD = 11.8) compared to Fall of 2023 (M = 98.70, SD = 3.26) for the Java Loops project, t(2) = 2.82, p = 0.0055. The students in Fall of 2023 (M =

96.56, SD = 7.13) also did significantly better on the Drawing App project compared to the Fall of 2022 students (M = 91.24, SD = 15.17), t(2) = 3.41, p = 0.00082. For the first submission of the Passport Class, the students in Fall of 2022 (M = 90.24, SD = 28.53) received significantly lower grades compared to the students in Fall of 2023 (M = 99.21, SD = 6.25), t(2) = 3.56, p = 0.00051. The last project that the students in Fall of 2023 (M = 97.65, SD = 8.41) received higher grades than the Fall of 2022 (M = 92.54, SD = 12.58) students was the PhotoManager project, t(2) = 0.0037, p = 0.49.

Table 5.10 Comparison of Projects between Fall of 2022 and Fall of 2023

Projects	**Fall 2022 CMSC 131 Average/ Standard Deviation**		**Fall 2023 CMSC131 Average/ Standard Deviation**		**T-statistic**	**P - Value**	**Cohen's d**
Java Loops	95.66	11.86	98.70	3.26	2.82	0.0055	0.35
Drawing App	91.24	15.17	96.56	7.13	3.41	0.00082	0.45
Passport Class (a)	90.24	28.53	99.21	6.25	3.56	0.00051	0.43
PhotoManager	92.54	12.58	97.65	8.41	2.95	0.0037	0.49

In Fall of 2023, the students did significantly better than the students in Fall of 2022 on the Debugging Quiz, Exam 2, Exam 3, and Final Exam, as seen in Table 5.11. There was a significant difference in the grades received in Fall of 2022 (M = 85.01, SD = 10.41) compared to Fall of 2023 (M = 96.19, SD = 6.47) for the Debugging Quiz, t(2) = 7.87, p <.001. The students in Fall of 2023 (M = 82.69, SD = 13.94) also did significantly better on the Exam 2 compared to the Fall of 2022 students (M = 76.37, SD = 17.32), t(2) =, p =. For Exam 3, the students in Fall of 2022 (M = 73.06, SD = 23.66) received significantly lower grades compared to the students in Fall of 2023 (M = 82.17, SD = 16.55), t(2) = 2.77, p = 0.0064. The students in Fall of 2023 (M = 84.71, SD = 13.63) received higher

grades than the Fall of 2022 (M = 77.25, SD = 20.16) students on their Final Exam, t(2) = 2.68, p = 0.0083.

Table 5.11 Comparison of Assessments between Fall of 2022 and Fall of 2023

Assessment	**Fall 2022 CMSC 131 Average/ Standard Deviation**		**Fall 2023 CMSC131 Average/ Standard Deviation**		**T-statistic**	**P - Value**	**Cohen's d**
Debugging Quiz	85.01	10.41	96.19	6.47	7.87	< .001	1.29
Exam 2	76.37	17.32	82.69	13.94	2.55	0.011	0.40
Exam 3	73.06	23.66	82.17	16.55	2.77	0.0064	0.45
Final Exam	77.25	20.16	84.71	13.63	2.68	0.0083	0.43

5.2 Neurodivergent and Neurotypical Student Experiences

This section aims to analyze data to answer the third and final research question: *What are the experiences of neurodivergent and neurotypical students after updating an introductory programming course to follow the universal design for learning (UDL) framework?* To answer this question, qualitative data from Fall of 2022 and Fall of 2023 survey answers and qualitative data from the interviews of the students in Spring of 2023 and Fall of 2023 were examined. The first section discusses the experiences of neurodivergent and neurotypical students with project descriptions before they had access to the project slides. Then, the neurodivergent and neurotypical students' experiences with the project slides are discussed. Lastly, neurodivergent and neurotypical student experiences with exams are discussed.

5.2.1 Student Experiences with Project Descriptions

Both Fall of 2022 and Spring of 2023 students mentioned that they had trouble understanding the project descriptions. In the surveys conducted in Fall of 2022, 4 responses mentioned that they had trouble understanding the project descriptions. Meanwhile, only 1 response stated that they understood the programming project descriptions. In the interviews conducted in Spring of 2023, 4 responses mentioned that they had trouble understanding the project descriptions. In the surveys conducted in Fall of 2022, 2 neurodivergent traited responses and 2 neurotypical traited responses mentioned that they had trouble understanding the project descriptions. Meanwhile, only 1 neurotypical traited response stated that they understood the programming project descriptions. In the interviews conducted in Spring of 2023, 2 neurodivergent traited responses and 2 neurotypical traited responses mentioned that they had trouble understanding the project descriptions. Therefore, both neurotypical and neurodivergent traited students had more trouble with understanding the project descriptions.

Neurodivergent traited students had a difficult time with understanding how the specifications of the project related to the code they needed to write. A Fall of 2022 neurodivergent traited student stated in a survey, "it takes a little more time than I'd like to admit to fully understand what I need to implement and learn." Neurodivergent students struggled with understanding how to do projects. In an interview, a Spring of 2023 neurodivergent traited student stated "I think figuring out how to do projects felt like a big learning curve for me because I felt like everyone else understood and I kind of just had to ask for help and I did and it ended up working out." Neurotypical traited students also struggled. In an interview, a neurotypical student expressed frustration with the descriptions of the programming projects. They stated, "Sometimes they are, they have, too much words more than they need. So it takes me time to read all the project… take[s] more time trying to

understand it and then to [understand] it's like, very basic idea that I did not require all the time and effort I put in to understand."

5.2.2 Student Experiences with Project Slides

In Fall of 2022 and Fall of 2023, the students were asked in each of the after-exam surveys, the short answer question: *Do you feel your programming assignment code reflects your understanding of the material?*

In Fall of 2022, 84.21% of neurotypical traited responses stated that their programming assignment code does reflect their understanding. In Fall of 2023, 83.33% of neurotypical traited responses stated that their programming assignment code does reflect their understanding.

In Fall of 2022, 60% of neurodivergent traited responses stated that their programming assignment code does reflect their understanding. In Fall of 2023, 76.93% of neurodivergent traited responses stated that their programming assignment code does reflect their understanding.

Table 5.12 Response Percentages for Neurodivergent and Neurotypical students that stated the programming assignment code reflects their understanding of the material

Do you feel your programming assignment code reflects your understanding of the material? (Yes)	**Fall 2022 CMSC 131**	**Fall 2023 CMSC 131**
Neurotypical Students	84.21%	83.33%
Total Neurotypical Student Responses	48	35
Neurodivergent Students	60%	83.33%
Total Neurodivergent Student Responses	12	10
All Students	77.92%	80.36%

Total Amount of Responses	60	45

In Fall of 2022, 14.06% of neurotypical traited responses stated that their programming assignment code somewhat reflects their understanding. In Fall of 2023, 14.29% of neurotypical traited responses stated that their programming assignment code somewhat reflects their understanding.

In Fall of 2022, 35% of neurodivergent traited responses stated that their programming assignment code somewhat reflects their understanding. In Fall of 2023, 15.38% of neurodivergent traited responses stated that their programming assignment code somewhat reflects their understanding.

Table 5.13 Response Percentages for Neurodivergent and Neurotypical students that stated the programming assignment code sometimes reflects their understanding of the material

Do you feel your programming assignment code reflects your understanding of the material? (Sometimes)	**Fall 2022 CMSC 131**	**Fall 2023 CMSC 131**
Neurotypical Students	14.04%	14.29%
Total Neurotypical Student Responses	8	6
Neurodivergent Students	35%	15.38%
Total Neurodivergent Student Responses	7	2
All Students	19.48%	14.29%
Total Amount of Responses	15	8

In Fall of 2022, 1.75% of neurotypical traited responses stated that their programming assignment code does not reflect their understanding. In Fall of 2023, 2.38% of neurotypical traited responses stated that their programming assignment code does not reflect their understanding.

In Fall of 2022, 5% of neurodivergent traited responses stated that their programming assignment code does not reflect their understanding. In Fall of 2023, 7.69% of neurodivergent traited responses stated that their programming assignment code does not reflect their understanding.

Table 5.14 Response Percentages for Neurodivergent and Neurotypical students that stated the programming assignment code does not reflect their understanding of the material

Do you feel your programming assignment code reflects your understanding of the material? (No)	**Fall 2022 CMSC 131**	**Fall 2023 CMSC 131**
Neurotypical Students	1.75%	2.38%
Total Neurotypical Student Responses	1	1
Neurodivergent Students	5%	7.69%
Total Neurodivergent Student Responses	1	1
All Students	2.60%	5.36%
Total Amount of Responses	2	2

Based on the experiences of the students in Fall of 2022 and Spring of 2023, project slides were created for the Fall of 2023 semester to help guide students through the programming projects. Both neurodivergent traited students and neurotypical traited students used the project slides when completing the assignments. In the surveys conducted, 3 neurodivergent tratied and 3 neurotypical traited responses mentioned that they had positive experiences with the project slides and 1 neurodivergent student had a bad experience with the project slides. In the interviews conducted, 3 neurotypical traited responses mentioned that they had positive experiences with the project slides and 1 neurodivergent traited student mentioned that they had negative experiences with the project slides. Therefore, neurotypical and neurodivergent students had positive experiences with the project slides.

Most of the students that responded to surveys and were interviewed had positive experiences with the slides. Students used the projects to help them understand what the project requirements were. Students used the slides over looking at the project web pages or the Javadoc. A neurotypical traited student stated in an interview, "the slide shows did help a lot. The one with the projects that they showed kind of showed you like examples and also the the images of what's supposed to be happening so that that actually did help a lot." When asked in a survey if the programming assignments were clear, another neurotypical traited student agreed with the previous student. They stated, "I think the assignment descriptions are very clear most times because it explicitly states what is expected from each method, but can be overwhelming seeing it all clumped together on the website. I prefer to look at the pdf slideshow to break the information from the website down and make it easier to process." Neurotypical students used the project slides because of the multitude of examples. Another neurotypical student stated in a interview, "I would always do my project with the slide set and not with the Java doc or with the descriptions they gave me because I felt like the slide set had like examples so it kind of made you understand like the different exceptions you're supposed to catch the different conditions, you're to test so that was the nice thing about the slides and I did use some slides quite frequently for my project." Therefore, the neurotypical students found the project slides more helpful than the web pages or the Javadoc.

Project 5, Photo Processing System, did not have slides created for the project. This was due to the project being created mid-semester. A neurotypical student mentioned that not having the slides made the project more difficult. They stated in an interview, "I think they were perfect the way they were like on the slides were helpful, but on some projects they didn't give us slides. They gave us like a Java doc and that was kind of hard." When

neurotypical students did not have access to the project slides they struggled with understanding what was being asked of them.

Not all students had only positive experiences with the slides. A neurodivergent student expressed in an interview that the slides did not help them enough. They stated, "I think I kind of like went slide by slide and use like use that to do all my projects. Yeah that I think it will still difficult… If they could actually like teach what code you use to write that, then that would be helpful." This student seemed to not just struggle with the requirements and specifications of the project, but the actual content of CMSC 131 instead. They are asking for the slides to teach them how to code, so they seem to be struggling with how concepts connect to each other. The project slides would not be able to show students how write the project code as this defeats the purpose of the assignment. However, this does show that neurodivergent students are still struggling with understanding the material.

This participant continuously expressed distress throughout the surveys and in the interview. When answering the survey question *Do you feel your programming assignment code reflects your understanding of the material? Why?* They stated in the first after-exam survey, "No, because honestly as a beginner doing all of this for the very first time everything is still very confusing. I try to follow the examples from class and all to do the programming assignments but I am still confusing about everything." In the second after-exam Survey they wrote, "For a beginner, everything is still very confusing and I have never done coding in my life so the programming assignments makes things more confusing instead of helping me learn step by step." In the third after-exam survey they wrote, "In a way yes because my programming assignment code is terrible and I don't understand the material at all." In the fourth after-exam survey they wrote, "I don't understand anything still because I am new to coding."

Some students also mentioned how to make improvements to the project slides to be more helpful. In a survey, a neurodivergent traited student asked for a table of contents in the slides and a student asked for more details on the project slides. In an interview, a neurotypical traited student asked for a table of contents in the slides and a neurodivergent traited student asked for more details on the project slides. Therefore, both neurotypical and neurodivergent students felt that a table of contents and more details on the slides would improve their experiences with the slides. These suggestions align with the UDL principles.

Adding a table of contents would increase the usability of the slides. A neurotypical traited student stated in an interview "the slides were definitely helpful… I wouldn't necessarily need all of them … I think that is good to have that many slides. But another good thing to have like a table or like a table of contents to see which slide was with which kind of topic." A neurodivergent traited student had a similar idea. They responded to the survey question *Do you feel the programming assignments' written descriptions are clear? Why?* with "For the first couple of projects I would say the instructions were clear. But the latest project (Project 4) the written descriptions were in a 294 slide pdf which was a hassle to look through. I wish there was another way to organize the information in a concise manner that isn't the java doc which was a bit too technical for me to use." Therefore, a table of contents would seem to help both neurodivergent and neurotypical students navigate the slides.

A neurotypical student expressed that they would have liked more detail when answering the same question. They stated, "Perhaps a more detailed pdf/video of the programming assignment would help to clear up any misunderstandings with the project." A neurodivergent traited student in an interview mentioned that the slides would be more helpful if they walked the student through actually writing the Java code instead of just to

understand the project requirements. They stated, "The PDF like helped you to understand the project. Bit by bit, but like for a beginner. You you don't know the language like you know, like, you know what you. You know what the project like needs to be and what the result needs to be, but you don't know how to actually write it in code." Adding more detail also aligns with UDL as it would increase the usability of the slides, but it may decrease the low physical effort of the slides. This is because there would be an increase in content which may be overwhelming to students. Therefore, one would have to be careful when adding more detail to the slides.

5.2.3 Student Experiences with Exams

In Fall of 2022 and Fall of 2023, the students were asked in the after-exam survey, the short answer question: Do you feel prepared for the exams? In Fall of 2022, 62.90% of neurotypical traited responses stated that they were prepared compared to 77.78% of Fall 2023 neurotypical students, as seen in 5.15. This implies that the neurotypical traited students felt more prepared in Fall of 2023. Similarly, Fall of 2023 neurodivergent students felt more prepared compared to Fall of 2022 students. In Fall of 2022, 55% of neurodivergent traited responses stated that they were prepared, while 66.67% of Fall 2023 neurodivergent traited students stated they were prepared. This could be due to the UDL improvements making it easier for both neurodivergent and neurotypical students to understand the material.

Table 5.15 Response Percentages for Neurodivergent and Neurotypical students that stated they felt prepared for the exams

Do you feel prepared for the exams? (Yes)	**Fall 2022 CMSC 131**	**Fall 2023 CMSC 131**
Neurotypical Students	62.90%	77.78%
Total Neurotypical Student Responses	39	35

Neurodivergent Students	55%	66.67%
Total Neurodivergent Student Responses	11	8
All Students	60.98%	75.44%
Total Amount of Responses	50	43

In Fall of 2022, 24.19% of neurotypical traited responses stated that they were sometimes prepared compared to 17.78% of Fall 2023 neurotypical students, as seen in Table 5.16. This implies that the neurotypical traited students felt more prepared in Fall of 2023. Similarly, Fall of 2023 neurodivergent students felt sometimes prepared less than Fall of 2022 students. In Fall of 2022, 20% of neurodivergent traited responses stated that they were sometimes prepared, while 8.33% of Fall 2023 neurodivergent traited students stated they were sometimes prepared. This could be due to the UDL improvements making it easier for both neurodivergent and neurotypical students to understand the material. This may have made students more confident about their exams.

Table 5.16 Response Percentages for Neurodivergent and Neurotypical students that stated they sometimes felt prepared for the exams

Do you feel prepared for the exams? (Sometimes)	**Fall 2022 CMSC 131**	**Fall 2023 CMSC 131**
Neurotypical Students	24.19%	17.78%
Total Neurotypical Student Responses	15	8
Neurodivergent Students	20%	8.33%
Total Neurodivergent Student Responses	4	1
All Students	23.17%	15.80%
Total Amount of Responses	19	9

In Fall of 2022, 12.9% of neurotypical traited responses stated that they were not prepared compared to 4.44% of Fall 2023 neurotypical students, as seen in Table 5.17. This

implies that the neurotypical traited students felt more prepared in Fall of 2023. However, Fall of 2023 neurodivergent students felt they were not prepared about the same as the Fall of 2022 students. In Fall of 2022, 25% of neurodivergent traited responses stated that they were sometimes prepared, while 25% of Fall 2023 neurodivergent traited students stated they were sometimes prepared. This could be due to the UDL improvements making it easier for neurotypical students to understand the material. This may have made students more confident about their exams. The UDL modifications were not enough to make neurodivergent students more confident in the material. This may suggest the UDL modification must also be applied to the exams themselves.

Table 5.17 Response Percentages for Neurodivergent and Neurotypical students that stated they did not feel prepared for the exams

Do you feel prepared for the exams? (No)	**Fall 2022 CMSC 131**	**Fall 2023 CMSC 131**
Neurotypical Students	12.90%	4.44%
Total Neurotypical Student Responses	8	2
Neurodivergent Students	25%	25%
Total Neurodivergent Student Responses	5	3
All Students	15.85%	8.77%
Total Amount of Responses	13	5

All three semesters of students expressed a need for the exam questions to be revised. In the surveys conducted in Fall of 2022, 3 neurodivergent traited responses and 9 neurotypical traited responses mentioned that the exam questions need to be revised to be clearer. In the interviews conducted in Spring of 2023, 1 neurodivergent traited student and 2 neurotypical traited students mentioned that the exam questions need to be revised to be clearer. In the surveys conducted in Fall of 2023, 1 neurotypical traited response mentioned

that the exam questions need to be revised to be clearer. In the interviews conducted in Fall of 2023, 1 neurotypical traited student mentioned that the exam questions need to be revised to be clearer.

In Fall of 2022, some students expressed that the exams did not reflect their knowledge of the material. A neurotypical traited student stated in a survey "there is a slight inconsistency between the assignment code and exam code. The pressure of exams may influence my ability to apply my understanding." Another neurotypical traited student stated, "The exams often include very specific tidbits which we went over once and are not that relevant to when one is actually creating a program. Especially as a non comp sci major, I am an applied math major, I feel this is true and kind of a problem with the course in general. Exams are too overemphasized when it should be more project, practical-based." Neurodivergent students also felt the exams did not reflect their knowledge of the material. Even though a neurodivergent traited student prepared for the exam, they did not feel confident about it. They stated, "I prepare and yet I never feel prepared enough for them." Another neurodivergent traited student had a similar experience. When taking the exam, they were asked questions that they were not prepared for. They stated, "some multiple choice questions catch me off guard but the code implementation sections are good". A neurodivergent traited student mentioned that the difficulty of the exam was harder than they anticipated even though they had prepared. They stated, "No. I completed 5 out of the 8 provided past exams provided for us to study, but the exam that we had was harder than all of the ones I did to practice." If exams are not showing the students knowledge, then the exams need to be revised. Exam questions that need to be improved significantly put neurodivergent students at a disadvantage. This is due to the common experience of neurodivergent students not being able to show their knowledge on traditional methods of assessment like exams.

In Spring of 2023, the students expressed a desire for the exam's coding questions to be more concise. A neurodivergent traited student in a interview stated, "sometimes like the written code at the very back it's just kind of a lump of words together and act like keep looking back at the instructions to figure out what I need to do. Like the clarifications and specific things don't really stick well in my brain… Like bullet points just to separate a little bit." When asked what could have been done differently to the exams to help, a neurotypical student stated, "I think less words could help." When asked the same question, a different neurotypical student stated, "maybe clearer descriptions and as well as maybe examples and like uh what do you call it? A condensed form of the project descriptions and instead of it all just being like a paragraph."

The students in Fall of 2023 also agreed that the coding questions should be concise. A neurotypical student expressed that the wording of the exams caused them to be confused. They stated, "But in 131, i think sometimes for the um exams the wording kind of confused me." After being asked what could have been done differently to the exams to help, the neurotypical student stated, "just going back in and making sure you check all the requirements sometimes can be a lot… i Would want like a i would want a box of like requirements."

Both neurotypical and neurodivergent students requested that the coding questions be revised to be more concise. This follows the UDL principle of low physical effort. If the questions are just a "lump of words together", it makes it more difficult for students to show their knowledge as they may overlook certain requirements. The students are also using a lot of time trying to understand the problem being asked. This minimizes the time they have to show they know the material.

Exams are only 50 minutes due to the class period being 50 minutes. All three semesters of students expressed a need for more time on the exams. In the surveys

conducted in Fall of 2022, 4 neurodivergent traited responses and 4 neurotypical traited responses mentioned they needed more time on the exams. In the interviews conducted in Spring of 2023, 1 neurodivergent traited student and 2 neurotypical traited students mentioned they needed more time on the exams. In the surveys conducted in Fall of 2023, 10 neurotypical traited responses mentioned they needed more time on the exams. In the interviews conducted in Fall of 2023, 1 neurotypical traited student mentioned they needed more time on the exams. Therefore, the neurotypical traited students mentioned needing more time compared to the neurodivergent traited students.

Both neurotypical and neurodivergent students struggled with finishing the exam in the time allowed. In Fall of 2022, a neurotypical traited student responded to the question, *Do you feel prepared for the exams?*, with "I feel prepared for the exams. However, I think that they are a little bit too long and slightly shorter exams could help me to actually revise the answers that I put down for the written coding exercises." Another neurotypical traited student responded to the same question with, "The time limit of 50 minutes is always daunting. Projects already take me quite a bit of time and having to code under a very short time constraint can be stressful. In some ways, the code we have to write is much shorter than what we do for projects but it still feels kind of daunting." Neurodivergent students often need more time to process questions. A neurodivergent student also struggled with the limited amount of time. They stated "I have more trouble thinking on my feet in situations like quizzes and exams. It's not that I don't understand the material and theory, I just can't think quickly like that… Like I said, I just can't think well in exam situations so despite preparedness I still don't do very well." This neurodivergent traited student experienced a struggle a lot of neurodivergent students have. The student may know the material well, but in an exam setting they aren't able to show that knowledge. However, both neurodivergent and neurotypical students would benefit from extra time on exams.

This follows the UDL principle Equitable Use as it is making the exams more accessible to students.

In Spring of 2023, students continued to express a need for more time on exams. A neurotypical traited student stated in an interview “one thing maybe the exams. I mean, the time it's it's not enough to do all the problems. So maybe, yeah, I, you know, I can solve all the problems but I do not have much time to do them.” They went on to explain, “So sometimes the exams are not the best reflections of the knowledge because it is a 50 minute timer box. Some people just are not good test takers or takes a time to like get acquainted into like what exactly is going on.” However, a neurotypical traited student stated that while in CMSC 131 they struggled with the exams, they were not struggling in CMSC 132. They stated in an interview, “I like the exams better now because I knew what to expect so like for these exams like last semester for 131. I wasn't finishing the exams. I wasn't really understanding like how to interpret it, but in 132, like, I was getting like A's and stuff on the exams, I was actually finishing them.” Most of the CMSC 132 students have experienced more exams that the CMSC 131 students. The students in CMSC 132 may have had better experience with the exams because they learned how to study and take the exams efficiently. Being able to code on paper is a different skill compared to being able to code in an IDE. The CMSC 132 students may have also acquired the ability to code on paper after having multiple exams to practice with.

In Fall of 2023, some students continued to express a need for more time on exams. A neurotypical traited student responded to the question, *Do you feel prepared for the exams?*, with “content wise I do and I like the way that the practice exams are made so available. In the exams though I feel like I usually am running out of time on the [coding] sections pretty regularly.” Another neurotypical student expressed a similar opinion when they were responding to the same question. They stated, “I feel prepared for the exams,

but an extra few minutes would be much more appreciated because I feel that my performance goes down when time is running out." Some students were stressed out by the time limit. A neurotypical traited student stated in an interview, "But when you do get to it then 50 minutes fly by quick and so that really stressed me out while I was taking an exam." There were no neurodivergent traited students that discussed an issue with the amount of time given for the exams. This is due to the small sample of neurodivergent students that were participating in Fall of 2023.

However, some students felt the time limit was appropriate. In a survey, 2 Fall of 2022 responses and 2 Fall of 2023 responses mentioned that they had enough time for the exams. In an interview, 3 Fall of 2023 students mentioned that they had enough time for the exams.In the surveys conducted in Fall of 2022, 1 neurodivergent traited response and 1 neurotypical traited response mentioned they had enough time on the exams. In the surveys conducted in Fall of 2023, 1 neurodivergent traited response and 1 neurotypical traited responses mentioned they had enough time on the exams. In the interviews conducted in Fall of 2023, 1 neurodivergent traited student and 2 neurotypical traited student mentioned they had enough time on the exams.

A neurotypical student in an interview after being asked if 50 minutes is enough stated, "Yeah I usually finish before the time went up." A neurodivergent traited student responded to the question, *Do you feel prepared for the exams?*, with "There is enough time I guess but I am still new to coding and very lost and confused so I don't feel prepared for the exams or any of the assignments." This student did not run out of time on the exam because they could not solve the exam questions. Therefore, it is possible that this student would want extra time if the student had a better understanding of the material.

In all three semesters, students expressed an annoyance with handwritten code. In the surveys conducted in Fall of 2022, 6 neurotypical traited responses mentioned they did

not like coding on paper. In the interviews conducted in Spring of 2023, 2 neurodivergent traited students and 1 neurotypical traited student mentioned they did not like coding on paper. In the interviews conducted in Fall of 2023, 1 neurotypical traited student mentioned they did not like coding on paper.

In Fall of 2022, students mentioned frustrations with coding on paper. A neurotypical student responded to the question, *Is there anything that would help you that was not provided to you?*, with "the exams, to me, seem like an attempt to shoe-horn computer science into a traditional exam style which does not work for me. This is why I am excited that [the instructor] might give us a take-home final which is sort of like a project, because that is actually how coding works! Not the on-paper, traditional, multiple-choice testing way." When responding to the question, *is there anything else that you would want people to know about how the course is structured?*, a neurotypical student stated "Assessments are handwritten. This is probably mentioned somewhere but it should be emphasized." No neurodivergent traited student in Fall of 2022 mentioned coding on paper for exams. It's possible that since neurodivergent students often struggle with exams, the students did not attribute negative or positive opinions about written assessments.

In Spring of 2023, students also expressed an annoyance with handwritten code. In an interview, a neurodivergent treated student stated "I remember going into like exam one and like, I was like it was weird because I had never done computer science on paper before and I didn't really. I don't know. It was weird… I actually have to figure out how to do exams on paper". Another neurodivergent traited student in an interview stated, "having to erase like big sections of code kinda sucks". Having to erase sections of code due to a mistake can be frustrating to students as it takes away time from answering questions. This goes against the tolerance for error UDL principle as it penalizes students for making mistakes.

Some students in Fall of 2023 expressed an annoyance with hand written code, but others were fine with writing code on paper. In an interview a neurotypical student stated, "In community college, their exams were on the computer… I liked the community college exams better… it was a really it was a difficult change… I wasn't very used to it cause like it's been a while like a really long time since I've written stuff on paper. And sometimes when I take the exams, my hands starts to hurt a little bit because I'm not used to it anymore." In contrast, another neurotypical traited student in an interview stated, "I was fine with uh writing on paper because I had to do that for computer science." A different neurotypical student stated in an interview, "It wasn't that different for me because, like I think, before each of the exams, I think we had a quiz. In the middle. And we had to code on paper… But sometimes like it was kind of like hard to remember, like the brackets or like the semicolon and things like that. And another thing, like when coding on paper is that you don't know the output." Neurotypical students may have been able to adapt to coding on paper. This puts neurotypical students at more of an advantage over neurodivergent students as neurodivergent students often struggle with adapting to different situations.

5.3 Other Student Needs and Ways to Respond to Them

In the after-exam surveys, the students were asked: *Is there anything that would help you that was not provided to you?* and *Is there anything else that you would want people to know about how the course is structured?*. These questions were added to gain additional insight in missing supports and unaddressed student needs. Even with the UDL additions, there is still improvement that can be made to the course. The students expressed that they wanted group projects, more examples and practice projects, solutions to exams and projects with adequate explanations, study guides, more help from teaching staff, slowing the pace of the course, and better placement for students' abilities to be assigned to correct courses, as seen in Table 5.18.

5.18 Missing supports and unaddressed student needs

Student Needs	**Fall 2022 CMSC 131 Percentage**	**Fall 2023 CMSC 131 Percentage**
Group Projects	4%	8.70%
More Examples and Practice Projects	16%	30.43%
Solutions to Exams and Projects	8%	13.04%
Study Guides	8%	8.70%
More Help from Teaching Staff	16%	4.35%
Slower Paced Course	40%	17.39%
Better Placement	8%	17.39%
Total Amount of Responses	12	12

Students expressed a want for group projects. Group projects do follow UDL principles, specifically the principle of a community of learners. A student in Fall 2022 wrote for the question, *Is there anything that would help you that was not provided to you?*, "Group projects would be fun." A student in Fall 2023 also seemed interested in group work when they wrote for the question, *is there anything else that you would want people to know about how the course is structured?*, "Lots of independent work, nothing in groups."

Students expressed a need for more examples and practice projects. This follows the UDL principle flexibility in use. A response to the survey question, *Is there anything that would help you that was not provided to you?,* from a Fall of 2022 student stated "Worksheets of just optional coding assignments would be nice so that I could practice in my spare time". A student from Fall of 2022 expressed that practicing coding is an absolute necessity: "Practice your code a lot. If you can, take this class on it's own or in a part-time semester, it's a huge commitment and yes, you can afford to not graduate on time it's okay."

The students believe that more practice and examples will help with their understanding. A student in Fall of 2023 stated, “I think that being given optional projects would help my understanding immensely”. In an interview Fall of 2023 student stated, “The code also was a good friend because I could actually interact with it, change stuff, experiment with it, and I feel like those really helped me take notes like understand that, do code examples and stuff.”

The students believe that examples help to understand how code flows together. A student in Fall of 2023 stated, “Perhaps more examples of the practicality of the code, like how it can be applied to other things.” In an interview a Fall of 2023 student stated, “Appreciated the examples but I sometimes was not good at imagining how things applied.”

Students expressed a need to practice coding and having examples that could help with different abilities. After being asked in an interview *if there are any resources that would have been helpful that were not provided for you?*, a Spring of 2023 student stated, they would like “More challenging examples, point to youtube to understand concepts, notes from previous students. Stuff for a visual learner”. A Fall of 2023 student stated “I don't know what exactly but more resources, websites, or videos maybe that actually help a person doing all this for the first time to actually learn coding step by step.”

Students expressed a need for solutions with thorough explanations to the exams and projects. In Fall of 2022, a student's response to the survey question, *is there anything that would help you that was not provided to you?*, stated “We are given previous exams and quizzes to study for upcoming quizzes, but there are no explanations for the answers - which is mostly fine but it would be helpful if they were explained on the solutions doc at least for the multiple choice”. In Fall of 2023, a student responded to the same question by stating “I think it would be nice if there were "answer keys" for the project assignments to

show students what the expected implementation of methods were or how to write efficient code (less lines and better style)."

Another way students asked for the flexibility in use principle to be used was when expressing a need for study guides for the course. In Fall of 2022 a student's response to the survey question *Is there anything that would help you that was not provided to you?* stated "A study guide/information sheet that combines all the topics learned throughout the semester". In Fall of 2022, a student responded to the same question by stating "Study guides with detailed notes that are also straight to the point".

Students expressed a need for more help from teaching staff. This implies that improvement needs to be made to the UDL principle, a community of learners. Students had mixed reactions about teaching assistants (TA) during discussion. In Spring of 2023, a student had a positive experience with their TA during discussion. They stated in an interview, "she was really good. She was able to break down a lot of questions and help out with projects. Her pacing was also good too and she was always welcome into an answering questions and going through the codes … even if we didn't want to do the worksheets and we want to focus on projects, she gave us that like agency of like what you you guys like to do." In Fall of 2023, a student stated in an interview that their TA really helped them with material. They stated, "the TA was excellent. He went over a lot and he made sure to explain concepts that were kind of down the line, but he made it sound like he made you taught it in a manner in which you knew the reason why things happened, but he didn't necessarily explain what they were because that's more advanced concepts."

Other students had negative experiences with their TA during discussion. In Spring of 2023, a student in an interview discussed an instance where the TA was having trouble teaching a concept. They stated, "she [TA] was trying to explain it and she got lost in the middle of discussion. And the thing is, is that that was one of the cases where it was like

close to before an exam. So the the people that were there were actually engaged because they wanted to understand this because they didn't understand it from before.” In Fall of 2023, a student in a survey stated “You may be stuck with a TA who doesn't teach very well or know the material (I don't want to assume they don't know; maybe rather they can't convey their knowledge well) and without prior knowledge you will be stuck teaching a lot of material to yourself, however I wouldn't say the pacing is such that being self taught is impossible.” In Fall of 2023, a student watched videos of other discussions to be more helpful. They stated in a survey, “I wasn't too fond of my TA, but after realizing I could also watch other TA's videos I found them much more engaging/helpful.” In Fall of 2023 a student expressed in an interview that the TA was not helpful because the full discussion time was not used. They stated, “In my experience for you know in in my ta discussions, we like most classes we left early because she because like she understands everything. So she just so she mainly zoomed through everything so we left classes like early, like most classes early, so I don't think it was structured well at all.”

Students also had mixed reactions about TAs during office hours. In Spring of 2023, a student had a positive experience. They stated in an interview, “they were super. Nice and helpful. They didn't really cause any stress.” However, a student in Spring of 2023 had a negative experience. They stated in an interview, “When I went to office hours, I had a bug that I said, maybe more than more than one whole day and trying to figure it out. And I decided to go to the office hours for the first time, but they wrote my name on the board and when my time came they said come another day.” A student in Fall of 2023 had a bad experience when they went to office hours twice for the same code. They stated in a interview, “It really kind of throws me off and also kind of like I get a little irritated cause I just spent some time writing this code because the TA helped me, but then go to another TA like a let's let's change this up a little bit or let's just do another different way”. Students

getting different answers from TAs can cause more confusion and frustration. It’s important that TAs are consistent, so the student can focus on learning the material.

Students expressed a need for the material to be taught at a slower pace. The UDL principle tolerance for error and flexibility in use can address this concern. Many students struggled to keep up with the course. In Fall of 2022, a student stated in a survey “Moves incredibly fast for people who've never coded before”. Another student in Fall of 2022 stated “It moves at an extremely fast pace and once you get lost it is hard to catch up.” In an interview a Spring of 2023 student stated, “I think they at the beginning, they went a little bit faster so it was kind of hard to keep up with because some of the things they were doing like they already knew it but like explaining it to us, it was like kind of harder for them to do.” In an interview a Fall of 2023 student stated, “It was like a lot of hard information, kind of very fast. And and the way like it was taught didn't really help because it was like a lot of information and then immediately a lot of projects and exams.”

Students in the Fall of 2022 and Fall of 2023 both mentioned in a survey that the assignments schedule is a lot of work. In Fall of 2022 a student stated, “Check the class web page and put all of the assignment dates on your calendar! It really does help so that nothing can sneak up on you at the last minute.” In Fall of 2023 a student stated, “I think a lot of people would appreciate that the class is recorded, but the weekly projects could be a lot for some people”

Another way students expressed a need for the flexibility in use and tolerance for error principles was when they asked for better placement of students in introductory programming courses. Students in all three semesters stated that there was a problem with the demographic that is meant to be taking the course vs who is actually taking the course. In Fall of 2022 a student wrote in a survey, “The course is structured so that there is a huge split between if you have done or not done programming before.” In Fall of 2023 a

neurotypical traited student wrote in a survey “I don't think exemption exams/CMSC133 [CMSC 133 is Object-Oriented Programming I for students that already have some programming experience] is pushed for enough. [The instructor] was quite condescending in the beginning of the year (any time someone asked a question that demonstrated their knowledge beyond the scope of the class so far he'd tell them they should be in 133) in regards to people taking 131 over 133, but throughout my whole process of orientation and such, 133 was never even mentioned to me as a class I could take.” In Fall of 2023, a student stated in an interview “I could tell most of the students had some prior experience in coding so I could tell that they understood everything. So it just made it more hard because I was like this is too difficult.” In Fall of 2023 a student wrote in a survey “it's just that for new students like me who actually don't have any prior experience doing any coding or computer programming learning all of this so fast in such a short amount of time can be very difficult and confusing. Taking a huge emotional and mental toll on us, especially those who are a bit hypersensitive like me.”

Students also seemed to have differences in opinions about whether or not the course was beginner friendly. In Spring of 2023, a student stated in an interview “I'll say first some people they said like it wasn't as beginner friendly for them… I don't know if it's like actually like an introduction, like introduction like babying you into it”. Another Spring of 2023 student stated, “I feel like they expect you to know a little bit, or at least to keep up with pace”. In Fall of 2023, a response from a student stated “I feel that this course is definitely not for beginners who have never done coding in their life. You definitely have to learn object-oriented java coding from somewhere else and fully understand and be able to do it before you take this course. Otherwise, you will definitely fail.” However, some students did find the course beginner friendly. In Fall of 2023, a student stated in an interview “This class is for a beginner, so if you're not a beginner you should do like the

next course or something. So yeah, I was pretty comfortable because [the instructor] made it clear that this was a beginner's course. And like the majority of the people in the class did not know how to code."

Chapter 6: Discussion & Conclusion

Chapter 6 summarizes the findings of this dissertation, discusses their significance and implications, and concludes this work by reviewing limitations and potential lines of future inquiry.. In the first section, the implications of the findings are discussed. The first section succinctly answers the three research questions and discusses how the modifications to the slides could be continued to be applied to other lecture slides, how to design supplementary material like the project slides could be recreated in other instances, how the updated lecture slides may have improved the student's experiences, how the project slides may have improved the student's experiences, how one might improve the exams to help future students, the implications of a majority of neurodivergent students not being registered at with UMD's disability Services, and how one might continue to improve introductory computer science courses based on student needs. Next the limitations of the study are discussed. These limitations include: students may not want to disclose their conditions, the RAADS-R not being a perfect measure for predicting whether students are neurodiverse, a small sample size, University of Maryland (UMD) competitive acceptances not reflecting all undergraduate populations, and constraints of the course. The conclusion section summarizes the dissertation and discusses what this work means for future work.

6.1 Implications of Findings

This chapter aims to answer the three research questions and discuss their implications. The three focal research questions are:

- How can universal design for learning principles be implemented into an introductory programming course's materials?
- How does updating an introductory programming course to follow the universal design for learning framework affect the experience of students?
- What are the experiences of neurodivergent and neurotypical students after updating an introductory programming course to follow the universal design for learning framework?

The first research question is answered and implications discuss how the modifications to the slides could be continued to be applied to other lecture slides and how to design supplementary material like the project slides could be recreated in other instances to further use Universal Design for Learning (UDL) Principles in other introductory programming courses. The second research question is answered by comparing Fall of 2022 Object-Oriented Programming I (CMSC 131) students with Fall of 2023 CMSC131 students. In order to answer this question the data that was analyzed were the grades and surveys. The third question is answered by the qualitative data from student surveys conducted in Fall of 2022 and Fall of 2023, student interviews conducted in Spring of 2023 and Fall of 2023, and TA interviews in the Spring of 2023 and the Fall of 2023.

6.1.1 Answer to Research Question 1

Universal Design for Learning (UDL) principles were implemented into an introductory programming course through modifications made to the lecture slides and designing project slides for the programming assignments. The modifications of the lecture slides and the design of the project slides were guided by the UDL principles: equitable use, flexibility in use, simple and intuitive, perceptible information, tolerance for error, low physical effort, size and space for approach and use, a community of learners, and instructional climate (Scott et al., 2003). Since some of the principles are not applicable to class material (e.g., tolerance for error, size and

space for approach and use, a community of learners, and instructional climate), only a subset of principles were used, including: equitable use, flexibility in use, simple and intuitive, perceptible information, and low physical effort.

Since the instructor agreed only to modifications to the format of the slides, the principle "Equitable use" was applied. Equitable use refers to instruction designed to be useful and accessible to all students. Since this study focused on neurodivergent students, the slides were modified to be more accessible to that population. To make sure that the slides were accessible to neurodivergent students, increasing readability was a main focus. This is because it would help students with dyslexia to read the slides, students with ADHD to pick up information quickly after inattentive episodes, minimize overstimulation by limiting the number of words and objects on the screen for autistic students, students with anxiety to find the information quickly in anxiety inducing events like a exam, and students with depression to learn information quickly after depressive episodes.

To increase the readability of the slides, built-in and automatic accessibility tools and manual changes were made to the slides. There are many different automatic tools built into software that one can use to check the accessibility of material. The lecture slides were created using Microsoft Powerpoint. Microsoft Powerpoint has an accessibility checker built into the software. The accessibility checker alerts the user to include alternative text objects and images, check that the contents on the slide is read in the correct order, check that the text and background colors have a sufficient contrast, and have every slide have a unique title tables are simple and don't contain split cells, and merged cells or nested tables (*Make Your PowerPoint Presentations Accessible to People with Disabilities - Microsoft Support*, n.d.). Most of these alerts are directed towards those with vision impairments, but they do affect readability. Therefore, even if one is trying to focus on neurodivergent students, it is still important for one to review the alerts and resolve them.

This tool is easy to use as the lower left corner has an accessibility button to bring the user to the Accessibility Checker. The lower left corner gives continuous feedback to the user about whether they need to investigate the results of the checker as they create or modify slides. The checker gives a short description of what the user needs to investigate and gives feedback for potential fixes. The accessibility checker gives an explanation to the user about why the accessibility warning or error can cause accessibility barriers for disabled users.

Another built in tool that can greatly help with readability is to use the spelling and grammar checker. Depending on the style of written content, slides may not follow proper sentence structure on all slides. There is also a possibility that technical terminology is not built into the spelling and grammar checker. Therefore, the built in spelling and grammar checker may highlight spelling and grammar errors that are not actually errors. It is very easy to ignore these warnings especially with pseudocode and Java syntax. However, it is still important for readability to check all warnings to make sure that misspellings and grammar mistakes are false alerts. While this may seem like an obvious step to create slides, often grammar and spelling mistakes are overlooked, especially as subject matter gets more technical.

Lastly, the Flesch Reading Ease Test was used. Reading ease tests are often not completely accurate because the ease is calculated using an algorithm and not based on the common language that is read and spoken. Therefore, when using a reading ease test, one must be informed on how the algorithm is calculating the score and whether that calculation is appropriate for the context it is being used in. The Flesch Reading Ease test calculates the score based on the number of syllables per word and words per sentence. Lecture slides need to be concise as they have a limited amount of space and need to be readable. The

Flesch Reading Ease Test is an appropriate measure to check how easy the slides are to read in this context as it is checking if the slides are concise and clear.

While the Flesch Reading Ease Test is not built into Microsoft Powerpoint, it is built into another common application Microsoft Word. An easy way to check the readability of the slides is to simply copy the text on the Powerpoint slides over to Microsoft Word. One can then get the Flesch Reading Ease Test score by going to File then options. A dialog box will then appear for the user to select proofing and show readability statistics. Microsoft Word will then bring the user to the readability statistics (*Get Your Document's Readability and Level Statistics - Microsoft Support*, n.d.). While this takes time, the steps are easy. The information received by taking these extra steps is very valuable and can quickly tell an instructor if their slides are too dense.

To further increase the readability of the slides, manual changes were made to the slides. These manual changes helped to increase readability by utilizing white space, formatting lists properly, checking the use of color, and changing the font. White space helps students to read and comprehend the slides as dense blocks of text can decrease comprehension (Initiative (WAI), 2024a). To add white space to each slide 6 pt. Spacing was added to all slides before and after each line of text and only one sentence was allowed on a single bullet. Both of these manual changes allowed for an increase in readability.

Another improvement to increase readability was to check that color was being chosen appropriately. One of the most important considerations when using color is to check that the text and the background have a strong contrast (Initiative (WAI), 2024b). This is a necessity for those with color deficiencies, but also helps all students read the text easily. The idea of strong contrast can also be expanded to include using different colors to highlight different aspects of text. Choosing to only use analogous colors can make it difficult to distinguish between two colors when skimming slides for important information

which decreases readability. It is also important to not use color sparingly to increase readability. If most of the slides are highlighted using different colors, it may be difficult to determine important information from other important information. Microsoft Powerpoint's Accessibility Checker is able to check if there is sufficient contrast between text and background colors (*Make Your PowerPoint Presentations Accessible to People with Disabilities - Microsoft Support*, n.d.). However, this tool is limited. Manual inspections still need to be done to ensure clarity.

Lastly, an improvement to increase readability can be to use sans-serif fonts instead of sans fonts. Which font is best for dyslexics and what is meant by best is often debated (Rello & Baeza-Yates, 2016). Most research on readable fonts focus on the speed in which the participants can read different fonts (Rello & Baeza-Yates, 2016). However, conditions like dyslexia do not just affect the speed at which a person can read. Reading comprehension is affected. Therefore, it is best practice to use fonts that are easy for everyone to read which are sans-serif fonts. In particular, the fonts recommended by dyslexia and low vision associations, Arial, Comic Sans, and Verdana, should be used to increase the readability of slides.

None of the modifications made to the slides were introductory programming or computer science specific. These modifications made slides more accessible by using built-in and automatic accessibility checking tools and manual modifications guided by the Web Content Accessibility Guidelines (WCAG) 2.1 (*Web Content Accessibility Guidelines (WCAG) 2.1*, 2023). All modifications could be applied to any slide packet for any course. The slides content can stay the same as only the format of the slides needs to be changed to increase readability. The drawback to applying these modifications is that they are very time consuming to implement to already existing slides. The built-in and automatic accessibility checking tools require manual inspection to correct issues. Therefore, in practice it is better

to apply these modifications directly into a slide packet when it is created. However, one can still implement the modifications to existing slides when given enough time.

When designing the project slides to complement the programming projects. The project descriptions on the website and Javadoc were hard to read and understand. The programming projects used the same text, but formatted in a more readable fashion. The project slides also offered more examples than the previous description on the website and Javadoc. The Universal Design for Learning principles that were used were equitable use, flexibility in use, the principles simple and intuitive, perceptible information, and low physical effort. Flexibility in use refers to accommodating all student abilities and providing them with multiple different methods of learning. Simple and intuitive refers to teaching in a straightforward way that eliminates as much unnecessary complexity as possible. Perceptible information refers to communicating necessary information in a way that would be effective for all students.

The simple and intuitive principle was applied when creating the project slides. The goal of the project slides was to give students a tool to help understand the project descriptions without showing them how to write the code. Simplistic English was used on the slides. This helped students to spend more mental effort on learning how their code worked and less mental effort on decoding project descriptions. To further minimize confusion, multiple examples were given to the students to help the students understand the project descriptions. While creating the project slides, the full project implementation code that was written as solutions to the projects was used to create multiple examples. The multiple examples were created by running different inputs into the project's methods and constructors. The different inputs were chosen based on the instructions of the project. The input, the calculations, and the output were then easily translated to the project slides.

Low physical effort, flexibility in use, simple and intuitive, and equitable use was applied to the creation of the project slides when deciding to use color to further help students understand the project descriptions. Low physical effort was applied to the project slides by copying the content found on the webpages and the JavaDocs to the project slides. This allowed the students to not have to look in multiple different locations to get all of the information they needed. Flexibility in use was applied by providing the students with multiple different methods of understanding the project descriptions. The project descriptions were explained using words, use of color and images and objects like tables to help students. Simple and intuitive was used when deciding to use consistent colors to help students recognize important parts of code like variables, if statements, and method calls. This also can help students further understand code as most integrated development environments use color to code syntax including Eclipse, the IDE the students used in the course to code (*Syntax Coloring*, n.d.). Equitable use was applied when picking the colors to check that students with color deficiencies would not be disadvantaged.

Choosing colors was not difficult. To choose colors that had an appropriate contrast, manual inspection was done using turning on grayscale. This is a built-in feature of Microsoft Powerpoint (*Make Your PowerPoint Presentations Accessible to People with Disabilities - Microsoft Support*, n.d.). When choosing colors, colors that looked different in gray scale were chosen. This meant that multiple colors could still be used, but the hue, brightness, and saturation was distinct enough to be distinguishable in grayscale.

Simple and intuitive, equitable use, and perceptible information was applied by giving the students access to the project slides in PDF and an animation in MP4 format. Since the lecture slides were given to the students in PDF format, giving the students access to the project slides in PDF format kept the material given to the students consistent. Giving

the students a PDF of the project slides allowed the students to use screen readers which applies equitable use. Perceptible information and Equitable use was further applied by giving the students the project slides in an animation MP4 format. By providing a MP4, the students would have control to pause, fast forward, rewind, speed up, and slow down the animation. This control allows all students to adjust the animation to their needs. The ability to adjust animations can benefit both neurotypical and neurodiverse students as it allows the students to have control over their learning. Neurodiverse and neurotypical students have slightly different needs. Therefore, the ability to customize learning can greatly improve the experiences of both sets of students instead of choosing a method geared to only one group.

This design process can be applied to other coding projects. To mimic the design process in other courses, the important aspects of the project slides are to not explicitly show code or pseudocode, use simplistic English, use color to highlight key parts of the code, use images and objects to visualize the code, be consistent across projects, provide multiple examples to students, and provide multiple formats to students. Applying the modifications that were made to the lecture slides as the project slides are being created can make the project slides more beneficial to students. This includes using built in and automatic accessibility tools, using the spelling and grammar checker, using the Flesch Reading Ease test, making use of white space, using best practices with lists, using best practices when using color, and using sans-serif fonts.

The modifications that were made to the lecture slides using built in and automatic accessibility tools took the least effort. The manual modifications took more effort as every slide had to be investigated separately. Changing the font and adding white space took the least amount of effort. It is easy to change the font as one can highlight the text and chase the font to a sans-serif.

Adding white space is a bit more complicated because more expertise with Microsoft Powerpoint is required. One can easily add white space to all slides by using the slide master. From the master slide, one can increase the white space between bullet points. Doing so may make content flow off the slide page. One can fix this by simply splitting the slides into two. A way white space can be added easily is to make sure every bullet only has one sentence. This spreads out the text which will increase the readability.

Choosing colors that are accessible are easy as Microsoft Powerpoint's Accessibility Checker will tell the user if the color contrast is not sufficient. One can also turn on the grayscale filter on Microsoft Powerpoint as a simple way to see if two colors are distinct enough in hue, brightness, and saturation. Changing the list format is also easy. One just needs to bullet the list and add a clear header to the list to increase readability. While this may seem obvious, one might choose to format lists in a different way to save space on the slides. However, this decreases the readability of the lists

Project slides in the future should also consider the students' need for a table of contents and more detail on the requirements. Adding the table of contents would not be difficult. The project slides were already broken up by class, constructor, and method, so one would simply just add where each class, constructor, and method appears in the slide packet. If one would like to link the table of contents to slides, it would have to be done separately. While time consuming it would not be difficult. Adding more detail would be difficult. It would not be acceptable to add pseudo code or java code to the project slides as that would take away from the project's goal. However, changing the text requirements could be possible. One would have to decide what parts of the requirements need more explanation. This can be difficult as the project slides can become cluttered quickly.

6.1.2 Answer to Research Question 2

To assess the experiences of the students after applying universal design for learning, the survey results and grades of the Fall of 2022 and Fall of 2023 students were analyzed. The data show students in Fall of 2023 had better experiences than the students in Fall of 2022. The Fall of 2023 students found the lecture slides to be more helpful than the Fall of 2022 students. While the Fall of 2022 students felt the lecture slides were only clear and easy to reference, the Fall of 2023 students felt the slides were clear, easy to reference, and easy to read. The Fall of 2022 students found the lecture slides to be harder to follow than the Fall of 2023 students. These results show that updating the lecture slides to follow UDL seemed to have made the content on the slides much easier to read as the updates were designed to increase the readability

The Fall of 2023 students used the slides after lecture more than the Fall of 2022 students. Updating the lecture slides to follow UDL seemed to have increased the usability of the slides when preparing for other aspects of the course like assignments, quizzes, and exams. The students may have used the slides more which led to them understanding the material on a deeper level. This could explain why the student in Fall of 2023 received higher grades on a quiz and 3 exams. The Fall of 2023 students felt the quizzes accurately showed how much they understood the material more than the Fall of 2022 students. This seems to suggest that the students in Fall of 2023 could show the knowledge they have on quizzes and exams easier than the Fall of 2022 Students.

The Fall of 2023 students mentioned there was a lack of examples on the slides more than the Fall of 2022 students. This may suggest that the students in Fall of 2023 may not have had to put as much effort into understanding the slides due to the updates which meant they could focus on other aspects that could have been improved like the lack of examples. Updating

the slides to follow UDL could have allowed for other weaknesses of the slides to be uncovered which could help improve the experiences of students in future semesters. As larger issues with the course materials are addressed, students are able to discuss ways to improve smaller issues.

The Fall of 2023 students felt the programming assignments show their ability to code better than the Fall of 2022 students. This could have been due to the project slides removing the problem the Fall of 2022 and Spring of 2023 students were having with understanding the project descriptions. Therefore, students may have been able to focus more on writing code. The students in Fall of 2022 were more confused about code compared to the Fall of 2023 students. The Fall of 2023 student also had more time to complete the assignments compared to the Fall of 2022 students. The students in Fall of 2023 seemed to have had a better understanding of the topics from the lecture due to the improvements of the slides before starting the programming assignments as seen by the Fall of 2023 students understanding memory maps and Java's interfaces more than the Fall of 2022 students.

The students in Fall of 2023 also had the supplementary project slides to guide them through the project requirements which could have also minimized confusion about the project requirements which could lead to more confidence about their own knowledge and ability to write code. The students in Fall of 2022 thought that the code was too difficult and had a bad structure more than the Fall of 2023 students. This may have led to the students having more time to complete the assignments compared to the Fall of 2022 students as they would not have had to spend as much time trying to figure out the requirements of the code. This could explain why the students had higher grades on 4 of the projects.

The Fall of 2022 students mentioned the programming assignments implemented topics from lecture more than the Fall of 2023 students. This could suggest that the students in Fall of

2023 had a better understanding of the topics from lecture due to the improvements of the slides before starting the programming assignments. They may not have needed to look back at the topics from the lecture as much to help them complete the programming assignments and would not find that an important aspect of the programming assignments. The students may have also spent more time with the supplementary slides and less time looking at the lecture slides to figure out how topics connect. They might not have thought about how topics individually fit while coding, but how code flows together. Adding supplementary slides to the programming slides could have been helpful to students as they can better connect the topics together.

The Fall of 2022 students mentioned the programming assignments improved understanding of the topics of the course. Students in Fall of 2023 may have had a better understanding of the topics from lectures due to the improvements of the slides before starting the programming assignments. This may be why the Fall of 2023 did not mention the programming assignments being catalysts for improving their understanding of the topics. This suggests updating the slides to UDL and adding supplementary slides to the programming slides were helpful to students because they were able to understand the code they were writing better.

Updating the lecture slides and adding material to the programming assignments may have helped students to do better on the programming assignments and exams which led to the Fall of 2023 students to have higher final grades. Updating the lecture slides to follow UDL and adding supplementary slides to the programming slides were helpful to students because they helped with understanding the topics of the course and helped them to be more successful in the course.

6.1.3 Answer to Research Question 3

To assess what the experiences of neurodivergent and neurotypicals students after updating the course to follow Universal Design for Learning principles, qualitative data from Fall of 2022 and Fall of 2023 survey answers and qualitative data from the interviews of the students in Spring of 2023 and Fall of 2023 were examined. In both Fall of 2022 and Spring of 2023, both neurodivergent traited and neurotypical traited students struggled with understanding the project descriptions. Neurodivergent and Neurotypical students explained in interviews and surveys that they had positive experiences with the slides. However, there was a neurodivergent participant that expressed negative experiences with the project slides.

This student seemed to have a distressing experience because they were not placed in the correct class. This student seemed to need a class before this one to help them understand programming at a much slower pace. Placing students in incorrect classes can harm students. They may believe that they are not 'smart' enough to be in a major. For neurodivergent students, placement is even more important. They may believe that it is impossible for students like them to succeed in the major. This can cause a larger issue, like computer science not having enough diversity. Without neurodivergent computer science majors, their unique experiences and thinking processes are not being utilized which can hinder innovation.

Unfortunately, CMSC 131 was not structured for students that need to be taught at a slower pace. Slower paced programming courses are available at UMD. However, students admitted to the CS program are not permitted to take these courses. This is an institutional barrier to neurodivergent students as often neurodivergent students learn at a slower pace due to traditional pedagogy not being compatible with how these students learn best. Neurodivergent students may need to take these classes first so they are primed for the

later classes, but CMSC131 is often the first course a student has to take for the major. So, often students that are struggling can be stuck in a situation where they cannot keep up with the work that is demanded of them. The student seemed to agree. In an interview they stated, "I can't believe that this is the first class that beginners should take for coding, cause that makes no sense like someone with no experience in coding takes this class for the first one, that makes no sense. I mean, this class definitely needs another class where you actually learn coding."

Taking a course that you are not ready to be in can be extremely damaging to students, especially neurodivergent students. In an interview, this student stated, "I mean, it was too overwhelming, but I guess I I learned some like some things because when you're when you're put into a pool of water and you don't know how to swim, you'll it is overwhelming, but you'll you'll learn something along the way, like you'll because you still have to survive in the water. So you'll try to. You know, swim like you don't know how, but you'll try to do something." The stress of not doing well in a course when a student is trying to succeed can be immense. Neurodivergent students often have a difficult time adjusting to undergraduate life, so to also be put in courses that you are not able to succeed in can cause students to give up on their major or even drop out.

The neurodivergent and neurotypical traited students that were prepared for the course, seemed to have benefited greatly from the modifications, but improvements to the design of the exams still seem necessary. In Fall of 2023, the neurotypical traited students felt they were more prepared for the exams than the neurotypical traited students in Fall of 2022. However, the same number of students said they did not feel prepared for the exams in both Fall of 2022 and Fall of 2023.

The universal design for learning principles cannot make all students succeed especially if they are not prepared for a course they are enrolled in. The principles cannot fix

aspects of the course that would benefit from revisions. In all three semesters, there were more neurotypical traited students than neurodivergent traited students that mentioned that the exam questions needed to be revised. It's a common experience for neurodivergent students to struggle with understanding exam questions. Therefore, the neurodivergent traited students may have not thought that the exam questions needed to be improved. Neurotypcial and neurodivergent traited students both struggled with the time limit for the exams. Neurotypical traited students mentioned the need for more time than the neurodivergent traited students. A common accommodation for neurodivergent students is to get extra time on exams. So, the neurodivergent students may not have thought to mention it as an important aspect that needs to be changed as they run out of time on exams regularly. Both Neurodivergent and neurotypical students expressed an annoyance with coding on paper. The students seem to believe that traditional exam styles for a programming course are not helpful. Students seemed to have to develop a separate skill for coding on paper. Neurodivergent students in particular struggle when environments change. What may seem like the knowledge students have in an IDE and transferring it to paper may lead to students not being able to show their knowledge as easily or as quickly because of the new environment.

The demographics survey showed that most of the neurodivergent students during the three semesters are not getting help through disability services, as seen in Table 6.1. Not all students participating in the study completed the demographics survey. 109 students completed the survey in Fall of 2022, 17 students completed the survey in Spring of 2023, and 19 students completed the survey in Fall of 2023. 2 students out of 52 students identified as neurodivergent traited disclosed that they were registered with the university’s disability services in Fall of 2022. 1 student out of 7 students identified as neurodivergent traited disclosed that they were registered with the university’s disability services in Spring

2023. 1 student out of 8 students identified as neurodivergent traited disclosed that they were registered with the university's disability services in Fall 2023. Therefore, only 31.48% of neurodivergent traited students are receiving accommodations.

Table 6.1The Number of Neurodivergent Traited Students Registered with Disability Services

	Number of students that took the Demographics survey	**Number of students who were identified as neurodivergent traited**	**Number of students registered with UMD's disability services**
Fall 2022	109	52	2
Spring 2023	17	1	7
Fall 2023	19	1	8

There is a huge lack of neurodivergent students registered with disability services in these courses. This is a problem as often neurodivergent students need accommodations to be able to succeed in courses. Using UDL when designing a course can help to support neurodivergent students, but they cannot support neurodivergent students fully. In the Fall of 2023 semester, the updated lecture slides and creation of the project slides could have minimized the need for certain accommodations like extensions on assignments. Only applying UDL to certain aspects of a course, like lecture slides and project slides, doesn't minimize challenges with other aspects of the course like exams. However, the professor did not feel comfortable with changes to the exams, so extra time on assessments could have been beneficial to these students. While universal design for learning has been shown to be helpful to all students and minimize the need for accommodations, UDL does not replace the need for accommodations.

6.2 Limitations

There were limitations to the study conducted. Students might not have wanted to disclose their neurodiverse condition(s). It's common for neurodivergent students to not want to disclose and mask their symptoms to appear neurotypical (Clouder et al., 2020; Hamilton & Petty, 2023). Therefore, neurodivergent students might also feel uncomfortable disclosing their condition(s) in the surveys.

This limitation was minimized by using the RAADS-R to place students in a neurodivergent traited group and a neurotypical traited group. However, the RAADS-R doesn't always show whether or not someone has neurodiverse traits. The RAADS-R can result in a false positive for autism in adults as the questions are not distinct enough to distinguish between other psychiatric disorders (Perinelli & Cloherty, 2023). In the case of this study, a false positive for autism due to other psychiatric disorders was a positive as the study was not studying any one specific neurodiverse condition, but neurodivergent students in general which includes all chronic psychiatric disorders. However, there is still a possibility that the score is a false positive and false negative for all neurodivergent conditions. Therefore, a student that was identified as neurodivergent traited could be neurotypical and vice versa.

The studies findings have the limitation of the sample size being small. Therefore, the findings may suggest the UDL modifications were more helpful then they would be if the population was the entire class. Students that were confident in their ability to succeed in the course may have been the students that decided to participate in the study. Therefore, the findings may suggest the UDL modifications were more helpful then they would be if the population was the entire class. Since Fall of 2022 had more students than Fall of 2023, it is possible that the students in Fall of 2023 did better than the students in Fall of 2022 because of them already being very good students. It is also possible that the Fall of 2023

students received better grades than the Fall of 2022 students due to differences in how teaching assistants (TA) graded material. This limitation is minimized during projects as the TAs only manually grade for style. Other parts of the grade are automated by having coded tests which award students points. However, the exams and quizzes are graded manually. Therefore, the Fall of 2022 students may have received lower grades due to harsher grading.

All students attending the same university is a limitation of the implications of the findings. Participants also might not reflect all other undergraduate student populations and other introductory classes of different subjects. The limited enrollment program and competitive acceptance at University of Maryland (UMD) limits the diversity of the population as the students have already been vetted to be relatively advanced students compared to other introductory programming students. There is a limitation due to the material being taught in Object-Oriented Programming I and Object-Oriented Programming II. This material might not be taught in all introductory programming courses. Therefore, students being taught with different material, programming language, and expectations might have different experiences than students at UMD. However, universal design for learning principles can be applied to any other programming courses as no UDL modification was content or language specific.

All proposed modifications could not be implemented due to the constraint of the structure of the course. This limitation was minimized by trying to discuss why changes could be beneficial to the students. However, the course instructor had authority to deny any modification that he was uncomfortable with.

6.3 Future Prospects

There is still work to be done to create a better environment for students. The students requested group projects, more examples and practice projects, solutions to exams and projects, study guides, more help from teaching staff, a slower paced course, and better placement. These suggestions may not work well with all types of students. Therefore, it is important to continue to study how these suggestions affect the experience of students.

Group projects may not be the best for all students, particularly neurodivergent students. In CMSC 132, the students did have a group project. In an interview, a neurodivergent traited student said "We had a group project at the beginning of the semester and I was with three dudes and I just felt like I wasn't really getting a word… I wish it wasn't limited to people in your sections, so you had to be in a group with… You could pick the people in your group. They just had to be in your section but I didn't know anyone in my section." Another CMSC 132 neurodivergent traited student said "I feel like I do really well in one-on-one conversations and I can hold my own very well there. But in large groups, I get more and more or I feel more and more introverted, or like, I feel like more and more nervous of talking and stuff." A common experience for neurodivergent students is to not engage during group work. Therefore while helpful to some students, group projects may not be beneficial to all students.

The teaching assistant (TA) for CMSC 132 saw students struggling with talking and working with their classmates during discussion. After being asked about what the TA would do if students were struggling with talking and working with other students, the TA stated, "I actually noticed some of that in my discussion sections because we would do like group work. So, I just, I kind of just tried to help encourage people to get into groups … [There was] one group in the front. They just, you know, they were sitting next to each other, but

they weren't talking. So I just went up and I tried to make a natural like I started talking to all of them … about the problem. And then, once you know, some ideas have been shared, let them go back and they keep talking about it to themselves, just trying to get the conversation started… They started working together more." If the TA hadn't intervened, it's possible that the students would not discuss the problem with each other. If the students are given group projects to work on, there is a chance that some students might be left out of the conversation because there wouldn't be a member of the teaching staff to intervene.

More examples and practice projects can help with their understanding, help them to understand how code flows together, and could help students with different abilities. However, more examples and practice problems could possibly have a negative effect on students. The students may become overwhelmed with the amount of examples and practice problems. This also may raise more concerns about students copying and pasting code instead of actually learning how to write code.

Study guides and thorough explanations to exam and project solutions may not be helpful to all students. Adding more material may be overwhelming for students. The students may only focus on material that is on study guides. This might make them weaker on other aspects of the course that the students may get confused from through explanations to exams and projects as the explanations may not correlate to their work.

The students asked for more help from the teaching staff. However, the students' experiences with the TAs may be a result of a larger problem. This could be because the TAs are not being supported. The Fall of 2023 TA explained how they did not receive any formal training. They stated, "I became a TA like my first week of school while walking to math class. So I didn't, I didn't go through any of the formal training… I like speed ran the training modules.a month or two after like I became a TA in like half an hour." This lack of support had led to a lack of consistency between TAs. The TA explained, " Depending on TA

you get you may or may not be screwed … Because it's like you want the grading to be fair, but it's like. I don't know. I feel like it was just really hard to make grading consistent across so many people because everyone has a mind of their own and like it's like."

The students asked for a slower paced course and better placement of students. It seems that there is a lack of understanding of who should be taking CMSC 131. Some students have felt that they needed to have come into the course with some programming knowledge to succeed. This goes against the instructors intention of this course being for students that haven't had programming experience before. This may have caused the courses policies, assignments, and assessments to not be appropriate for the students that are advertised to take the course.

6.4 Conclusion

An increase of neurodivergent students pursuing higher education and computer science degrees has resulted in a need for improving the way we teach computer science courses to help students learn the material while minimizing the struggles of neurodivergent students (Aguar et al., 2016; Clouder et al., 2020). Colleges and universities are legally bound to provide accommodations to students with documented disabilities, accommodations are often not enough (W. M. Hadley, 2007).

Computer science skills are important across STEM fields and also in many non-STEM related fields (Koushik & Kane, 2019). A lack of support for neurodivergent students in higher education has resulted in neurodivergent students being underrepresented in STEM fields (Dunn et al., 2012). There is a growing demand for computer science skills (Koushik & Kane, 2019). Programming courses have high dropout rates due to them being regarded as difficult (Robins et al., 2003). To improve retention, programming courses need to be accessible to as many students as possible (Hansen et al., 2016).

This dissertation presented a study conducted to investigate ways to make introductory programming courses more supportive to students by using the Universal Design for Learning Framework. The study used UDL in order to make modifications to help neurodivergent students. In particular, five neurodiverse conditions were focused on ADHD, ASD, Dyslexia, Depression, and Anxiety.

The study pursued the following questions:

- How can universal design for learning principles be implemented into an introductory programming course's materials?
- How does updating an introductory programming course to follow the universal design for learning framework affect the experience of students?
- What are the experiences of neurodivergent and neurotypical students after updating an introductory programming course to follow the universal design for learning framework?

The data from the surveys revealed that Fall of 2023 students understood topics better, felt the quizzes accurately showed how much they understood the material more, and found the slides more helpful than the Fall of 2022 students. The grades showed that the students in Fall of 2023 receive higher grades on four projects, a debugging quiz, three Exams, and their final grade in the course compared to the students in Fall of 2022. The survey revealed that students expressed a need for wanting group projects, more examples and practice projects, solutions to exams and projects with adequate explanations, study guides, more help from teaching staff, slowing the pace of the course, and better placement for students' abilities to be assigned to correct courses.

There are still many improvements to be made on how we can improve introductory programming courses. There may be other ways that universal design for learning principles can be implemented into courses. It's important to study other ways of implementing these

principles to see what the best method is. When discussing improvements that can be made to courses, it's important to focus on minority groups of students like neurodivergent students to make sure that any additions are not detrimental to a student's learning even if it helps the majority like neurotypical students. It will also be important to consider the burden on instructors to implement the principles. More research is needed about applying UDL to introductory programming courses. This research could help inform how instructors can design better environments for students. A better environment for all students will create better and more diverse computer science professionals.

Appendices

Appendix A

CMSC131 (Fall 2022, Nelson's Sections 01*, 03*) Object-Oriented Programming I

Syllabus

Introduction

This is the first programming course for Computer Science majors with a focus on object-oriented programming. The goal of the course is to develop skills such as program design and testing as well as the implementation of programs using a graphical IDE. All programming will be done in Java.

Prerequisites

Corerequisite: Math140

Coordinator

Nelson Padua-Perez, Office: IRB 2210

Textbook

No required textbook.

Class Format

- Lectures and lab/discussion sessions be recorded (if this represents a problem for you contact your instructor).
- No pop quizzes.

- Lecture and lab/discussion session attendance is NOT required, however, you are responsible for any material covered in lecture and lab/discussion session.
- You do not need to notify your instructor if you will be missing lecture or lab/discussion session, unless graded material (e.g., exam) takes place on that particular lecture.
- We may have a limited number of online office hours in addition to on-campus office hours.
- Students are expected to abide by the university health guidelines.
- Email should be used for urgent matters and not to address project questions, lecture material questions, etc.
- The above format will be updated in case the class moves to an online format.
- All programming assignments will be individually developed, unless specified otherwise.

Course Topics (Subject to Change)

Intro to Computer Systems	Variables, Operators
Expressions, Statements, Methods	Java Text Input/Output
Conditionals	Loops
Principles of Object Oriented Programming	Basics of Program Design
Testing and Debugging	Java Memory Map
Arrays and Java ArrayLists	Java interfaces

Inheritance Overview	Recursion

Grading

Programming Assignments (Projects), Exercises, Lab Work	40%
Quizzes	3%
Semester Exams (3), (13%, 16% and 16%)	45%
Final Exam	12%

Grading Concerns

It is your responsibility to submit regrade requests by a specified deadline; no regrade requests will be processed afterwards. Deadlines to address any grading concerns will be available at Grading Concerns.

Assignments

- Deadlines - All assignments are due at 11:55 pm and you have until 11:55 pm of the next day to submit your work with a 12% penalty. You will not receive any credit for a submission after the late deadline. The submit server will use 11:56 pm as the deadline, otherwise assignments submitted exactly at 11:55 pm will be considered late. The actual deadline for assignments is 11:55 pm. One minute late is considered a late submission.
- Submit Server - You need to use the submit server to submit you work. We will not accept work submitted otherwise (e.g., email, etc.). You need to make sure that your assignment solution works in the submit server, otherwise you may lose most of the credit.

- Which Assignments Gets Graded - For programming assignments the one with public/release/secret tests that scores the highest in the submit server after a late penalty (if any) has been applied. We only use public/release/secret tests scores to select the submission to grade. Other assignment requirements (e.g., style, methods you must implement, allowed classes, etc.) are not considered. We will evaluate those requirements on the selected submission. We cannot select a particular submission to grade. We will grade the highest scoring submission as described above. Keep in mind you may get a total score lower than expected if the highest scoring submission is missing requirements a lower scoring submission satisfies.

- Closed Assignments - All programming assignments in this course are to be written individually (unless explicitly indicated otherwise).You may discuss assignments only with TAs, instructor or via Piazza.

- No Pop Quizzes/Pop Lab Work - There are no pop quizzes nor pop lab exercises. We will announce in advance (at least one day) if there is any work you need to submit for a grade.

- Projects and exercises are posted by 6 pm on the day specified on the schedule.

Regarding Posting of Assignments' Solutions/Implementations

- Posting of any assignment solution (even after the course is over) in a publicly available online location (e.g., github, Chegg) is prohibited under the Code of Academic Integrity (facilitation of academic dishonesty). Any student responsible for publicly posting assignments' solutions will be reported to the Office of Student Conduct and risks the sanction of an "XF" in the course.

- Posting of your assignments in a private repository where only selected people (e.g., potential employers) have access is OK.

Covid Related

- The campus facilities division have put QR codes on desks throughout the campus. A student can scan the QR code where they are sitting each class, which will "register" their place and location at a particular time. Additional information at https://provost.umd.edu/node/4243 (Q&A 4).

TA Room/Office Hours

Office hours get extremely busy the day before an assignment deadline and getting help is not guaranteed. Please start your assignments early so you can address any problems during office hours.

Backups

You are responsible for creating backups of your work using any approach (make sure your work is not accessible to others). You are required to submit your work to the submit server often, so you have a backup copy. No extensions will be granted if you lose your work and you had no submit server backups.

Piazza

We will be using (Piazza) for class communication. You will not be able to register to Piazza yourself. Your instructor will register you using the email address you have in the school system. Posting of any kind of code in Piazza is not allowed.

Class Announcements

You are responsible for checking announcements (at least twice a day) we post in the announcements Piazza folder. An old announcements Piazza folder will have old announcements.

Excused Absence and Academic Accommodations

See the section titled "Attendance and Missed Assignments" available at Course Related Policies.

Accessibility

See the section titled "Accessibility" available at Course Related Policies.

For Accessibility & Disability Students

If you are an ADS (https://counseling.umd.edu/ads) student (others ignore this information).

ADS students: you are responsible for reserving a space at ADS to take exams (we cannot provide that support). Keep in mind ADS has deadlines regarding by when to schedule a day/time to take exams. If your main accommodation is extra time in exams and quizzes, you don't need to meet your instructor (just bring any form that needs a signature to lecture).

Academic Integrity

Please read this information carefully. We take academic integrity matters seriously.

1. Academic dishonesty includes not only cheating, fabrication, and plagiarism, but also includes helping other students commit acts of academic dishonesty by allowing them to obtain copies of your work. All submitted work must be your own. Cases of

academic dishonesty will be pursued to the fullest extent possible as stipulated by the Office of Student Conduct.

2. Situations that often lead to academic integrity violations:
    - A student's friend/roommate shares an assignment's code. Once you provide your code to another student, you are a facilitator, even if you indicate to the student "not to copy-paste" any of it. Actually we had a case in which a student CS degree was revoked for this reason.
    - Students use online resources (github, Chegg, etc.) to find assignments' solutions. The solutions are found by several students and all will be involved in an academic case.
    - Students assume we don't have tools that check for similarities among all students' submissions.
    - Students get desperate and don't want a 0 in the assignment.
    - Students are not aware of the expectations regarding academic integrity.
    - Students assume we don't take academic integrity matters seriously.
    - You should only receive assistance from instructors/TAs. We have seen cases in which the use of tutors have led to academic integrity violations (e.g., tutors looked for assignment's solutions online).
3. The Office of Student Conduct is responsible for handling academic integrity matters. After a report is submitted by an instructor, the case is evaluated by the office and it could result in an XF grade, degree revocation, or dismissal from the university.

4. One of the most negative consequences of academic integrity violations is the emotional burden an academic integrity case has on a student. We have seen students extremely distraught as a result of an academic integrity violation. In many cases students chances for recommendations, TA positions, and other opportunities are negatively affected.

5. Please read the section titled "Academic Integrity" available at Course Related Policies and the information available at Academic Integrity

Class Concerns

If you or your parents have any class concerns, feel free to contact the instructor. If an issue arises with the instructor, report it using the form available at https://www.cs.umd.edu/classconcern.

Miscellaneous

- We only use ELMS for videos.
- At the end of the semester visit (www.courseevalum.umd.edu) to complete your course evaluations.
- Contact your instructor, the Counseling Center , or both, if you are experiencing difficulties that affect your performance in your courses.
- UMD Course related policies are available at http://www.ugst.umd.edu/courserelatedpolicies.html.
- We plan to record lectures and lab/discussion sessions, but technical problems may prevent us from creating a recording. You are still responsible for any material covered in lecture and lab/discussion session.

Copyright

Appendix B

CMSC132 (Spring 2023, Nelson's Sections 01*) Object-Oriented Programming II Syllabus

Introduction

Object-Oriented Programming II covers the design, building, testing, and debugging of medium-size software systems. Students will learn object-oriented methodology, algorithms, and data structures, to create effective and efficient problem solutions. All programming will be done in Java.

Prerequisites

Prerequisite → Minimum grade of C- in CMSC131; or must have earned a score of 5 on the A Java AP exam; or must have earned a satisfactory score on the departmental placement exam. In addition, a minimum grade of C- in MATH140 (Calculus I).

Credits

Credits → 4

Coordinator

Nelson Padua-Perez, Office: IRB 2210

Textbook

No required textbook. Information about a recommended textbook by Carrano and Henry, that you do not need to buy to be successful in the course, can be found below. Some online textbooks you could use can be found at Introduction to Programming Using Java, Eighth Edition and Think Java 2e (look for "Download Think Java in PDF").

Title	Authors	ISBN	Type
Data Structures & Abstractions with Java, 5th Edition	Carrano, Henry	9780134831695	Recommended

Class Format

- Lectures and lab/discussion sessions be recorded (if this represents a problem for you contact your instructor).
- No pop quizzes.
- Lecture and lab/discussion session attendance is NOT required, however, you are responsible for any material covered in lecture and lab/discussion session.
- You do not need to notify your instructor if you will be missing lecture or lab/discussion session, unless graded material (e.g., exam) takes place on that particular lecture.
- We may have a limited number of online office hours in addition to on-campus office hours.
- Students are expected to abide by the university health guidelines.
- Email should be used for urgent matters and not to address project questions, lecture material questions, etc.
- All programming assignments will be individually developed, unless specified otherwise.

  Course Topics (Subject to Change)

- Object-oriented software development

    - Software life cycle
    - Requirements & specifications
    - Designing classes
    - Testing & code coverage
    - Design patterns
- Algorithms & data structures
    - Asymptotic efficiency
    - Lists, stacks, queues
    - Trees, heaps
    - Sets, maps, graphs
    - Recursion
- Programming skills
    - Inheritance in Java
    - Java collection framework
    - Threads, synchronization
    - Exceptions

Grading

Programming Assignments (Projects), Exercises, Lab Work	38%

Quizzes	4%
Semester Exams (3), (14%, 16% and 16%)	46%
Final Exam	12%

Grading Concerns

It is your responsibility to submit regrade requests by a specified deadline; no regrade requests will be processed afterwards (even if the are grading errors). If you don't address a grading concern by the specified deadline, we will assume you have reviewed the graded work and are satisfied with your current grade. Deadlines to address any grading concerns will be available at Grading Concerns.

Assignments

- Deadlines - All assignments are due at 11:55 pm and you have until 11:55 pm of the next day to submit your work with a 12% penalty. You will not receive any credit for a submission after the late deadline. The submit server will use 11:56 pm as the deadline, otherwise assignments submitted exactly at 11:55 pm will be considered late. The actual deadline for assignments is 11:55 pm. One minute late is considered a late submission.

- Submit Server - You need to use the submit server to submit you work. We will not accept work submitted otherwise (e.g., email, etc.). You need to make sure that your assignment solution works in the submit server, otherwise you may lose most or all of the assignment credit.

- Which Assignments Gets Graded - For programming assignments the one with public/release/secret tests that scores the highest in the submit server after a late

penalty (if any) has been applied. We only use public/release/secret tests scores to select the submission to grade. Other assignment requirements (e.g., style, methods you must implement, allowed classes, etc.) are not considered. We will evaluate those requirements on the selected submission. We cannot select a particular submission to grade. We will grade the highest scoring submission as described above. Keep in mind you may get a total score lower than expected if the highest scoring submission is missing requirements a lower scoring submission satisfies.

- Closed Assignments - All programming assignments in this course are to be written individually (unless explicitly indicated otherwise).You may discuss assignments only with TAs, instructor or via Piazza.

- No Pop Quizzes/Pop Lab Work - There are no pop quizzes nor pop lab exercises. We will announce in advance (at least one day) if there is any work you need to submit for a grade.

- Projects and exercises are posted by 6 pm (or earlier) on the day specified on the schedule.

Regarding Posting of Assignments' Solutions/Implementations

- Posting of any assignment solution (even after the course is over) in a publicly available online location (e.g., github, Chegg) is prohibited under the Code of Academic Integrity (facilitation of academic dishonesty). Any student responsible for publicly posting assignments' solutions will be reported to the Office of Student Conduct and risks the sanction of an "XF" in the course.

- Posting of your assignments in a private repository where only selected people (e.g., potential employers) have access is OK.

TA Room/Office Hours

Office hours get extremely busy the day before an assignment deadline and getting help is not guaranteed. Please start your assignments early so you can address any problems during office hours.

Backups

You are responsible for creating backups of your work using any approach (make sure your work is not accessible to others). You are required to submit your work to the submit server often, so you have a backup copy. No extensions will be granted if you lose your work and you had no submit server backups.

Piazza

We will be using (Piazza) for class communication. You will not be able to register to Piazza yourself. Your instructor will register you using the email address you have in the school system. Posting of any kind of code in Piazza is not allowed.

Class Announcements

You are responsible for checking announcements (at least twice a day) we post in the announcements Piazza folder. An oldannouncements Piazza folder will have old announcements.

Excused Absence and Academic Accommodations

See the section titled "Attendance and Missed Assignments" available at Course Related Policies.

Accessibility

See the section titled "Accessibility" available at Course Related Policies.

For Accessibility & Disability Students

If you are an ADS (https://counseling.umd.edu/ads) student (others ignore this information).

ADS students: you are responsible for reserving a space at ADS to take exams (we cannot provide that support). Keep in mind ADS has deadlines regarding by when to schedule a day/time to take exams. If your main accommodation is extra time in exams and quizzes, you don't need to meet your instructor (just bring any form that needs a signature to lecture).

Academic Integrity

Please read this information carefully. We take academic integrity matters seriously.

1. Academic dishonesty includes not only cheating, fabrication, and plagiarism, but also includes helping other students commit acts of academic dishonesty by allowing them to obtain copies of your work. All submitted work must be your own. Cases of academic dishonesty will be pursued to the fullest extent possible as stipulated by the Office of Student Conduct.
2. Situations that often lead to academic integrity violations:
    - A student's friend/roommate shares an assignment's code. Once you provide your code to another student, you are a facilitator, even if you indicate to the student "not to copy-paste" any of it. Actually we had a case in which a student CS degree was revoked for this reason.

- Students use online resources (github, Chegg, etc.) to find assignments' solutions. The solutions are found by several students and all will be involved in an academic case.
- Students assume we don't have tools that check for similarities among all students' submissions.
- Students get desperate and don't want a 0 in the assignment.
- Students are not aware of the expectations regarding academic integrity.
- Students assume we don't take academic integrity matters seriously.
- You should only receive assistance from instructors/TAs. We have seen cases in which the use of tutors have led to academic integrity violations (e.g., tutors looked for assignment's solutions online).

3. The Office of Student Conduct is responsible for handling academic integrity matters. After a report is submitted by an instructor, the case is evaluated by the office and it could result in an XF grade, degree revocation, or dismissal from the university.
4. One of the most negative consequences of academic integrity violations is the emotional burden an academic integrity case has on a student. We have seen students extremely distraught as a result of an academic integrity violation. In many cases students chances for recommendations, TA positions, and other opportunities are negatively affected.
5. Please read the section titled "Academic Integrity" available at Course Related Policies and the information available at Academic Integrity

Class Concerns

If you or your parents have any class concerns, feel free to contact the instructor. If an issue arises with the instructor, report it using the form available at https://www.cs.umd.edu/classconcern.

Miscellaneous

- We only use ELMS for videos.
- At the end of the semester visit (www.courseevalum.umd.edu) to complete your course evaluations.
- Contact your instructor, the Counseling Center , or both, if you are experiencing difficulties that affect your performance in your courses.
- UMD Course related policies are available at http://www.ugst.umd.edu/courserelatedpolicies.html.
- We plan to record lectures and lab/discussion sessions, but technical problems may prevent us from creating a recording. You are still responsible for any material covered in lecture and lab/discussion session.

Copyright

Appendix C

CMSC131 (Fall 2023, Pedram's and Nelson's Sections) Object-Oriented Programming I

Syllabus

Introduction

This is the first programming course for Computer Science majors with a focus on object-oriented programming. The goal of the course is to develop skills such as program design and testing as well as the implementation of programs using a graphical IDE. All programming will be done in Java.

Prerequisites

Corerequisite: Math140

Coordinator(s)

Pedram Sadeghian, Office: IRB 2214

Nelson Padua-Perez, Office: IRB 2210

Textbook

No required textbook. Some online textbook references can be found at Resources.

Class Format

- Lectures and lab/discussion sessions be recorded (if this represents a problem for you contact your instructor). Keep in mind that technical problems may prevent us from creating a recording. You are still responsible for any material covered in lecture and lab/discussion session.

- No pop quizzes.
- Lecture and lab/discussion session attendance is NOT required, however, you are responsible for any material covered in lecture and lab/discussion session.
- You do not need to notify your instructor if you will be missing lecture or lab/discussion session, unless graded material (e.g., exam) takes place on that particular lecture.
- We may have a limited number of online office hours in addition to on-campus office hours.
- Pedram and Nelson are co-teaching and they will have the same projects, quizzes, exams, etc.

Email Policy

Email (to both instructors and TAs) should be used for urgent matters and not to address project questions, lecture material questions, etc. Due to the large number of students in the class (close to a 1000 :)) email should be used only when necessary.

Course Topics (Subject to Change)

Intro to Computer Systems	Variables, Operators
Expressions, Statements, Methods	Java Text Input/Output
Conditionals	Loops
Principles of Object Oriented Programming	Basics of Program Design
Testing and Debugging	Java Memory Map

Arrays and Java ArrayLists	Java interfaces
Inheritance Overview	Recursion

Grading

Programming Assignments (Projects), Exercises, Lab Work	39%
Quizzes	3%
Semester Exams (3), (10%, 16% and 16%)	42%
Final Exam	16%

Grading Concerns

It is your responsibility to submit regrade requests by a specified deadline; no regrade requests will be processed afterwards (even if the are grading errors). If you don't address a grading concern by the specified deadline, we will assume you have reviewed the graded work and are satisfied with your current grade. Deadlines to address any grading concerns will be available at Grading Concerns.

Assignments

- Deadlines - All assignments are due at 11:55 pm and you have until 11:55 pm of the next day to submit your work with a 12% penalty. You will not receive any credit for a submission after the late deadline. The submit server will use 11:56 pm as the deadline, otherwise assignments submitted exactly at 11:55 pm will be considered late. The actual deadline for assignments is 11:55 pm. A submission one minute late is considered a late submission.

- Submit Server - You need to use the submit server to submit you work. We will not accept work submitted otherwise (e.g., email, etc.). You need to make sure that your assignment solution works in the submit server, otherwise you may lose most of the credit.

- Which Assignments Gets Graded - For programming assignments the one with public/release/secret tests that scores the highest in the submit server after a late penalty (if any) has been applied. We only use public/release/secret tests scores to select the submission to grade. Other assignment requirements (e.g., style, methods you must implement, allowed classes, etc.) are not considered. We will evaluate those requirements on the selected submission. We cannot select a particular submission to grade. We will grade the highest scoring submission as described above. Keep in mind you may get a total score lower than expected if the highest scoring submission is missing requirements a lower scoring submission satisfies.

- Closed Assignments - All programming assignments in this course are to be written individually (unless explicitly indicated otherwise).You may discuss assignments only with TAs, instructor or via Piazza.

- No Pop Quizzes/Pop Lab Work - There are no pop quizzes or pop lab exercises. We will announce in advance (at least one day) if there is any work you need to submit for a grade.

- Projects and exercises are posted by 6 pm on the day specified on the schedule.

AI (Artificial Intelligence) Tools Usage

The use of AI (Artificial Intelligence) tools (e.g., ChatGPT, Bing AI) for the completion of graded work (e.g., programming assignments) is not allowed and represents an academic integrity violation (see information below).

Regarding Posting of Assignments' Solutions/Implementations

- Posting of any assignment solution (even after the course is over) in a publicly available online location (e.g., github, Chegg) is prohibited under the Code of Academic Integrity (facilitation of academic dishonesty). Any student responsible for publicly posting assignments' solutions will be reported to the Office of Student Conduct and risks the sanction of an "XF" in the course.
- Posting of your assignments in a private repository where only selected people (e.g., potential employers) have access is OK.

Office Hours

Office hours get extremely busy the day before an assignment deadline. Help during office hours is not guaranteed. TAs/instructors cannot stay holding office hours after the office hours period ends because students are waiting for help (this applies to online/virtual office hours). The sooner you start working on a project, the better your chances of getting help. Please, leave the TA room once you have received help and do not use the TA Room as a working area.

Backups

You are responsible for creating backups of your work using any approach (make sure your work is not accessible to others). You are required to submit your work to the submit server often, so you have a backup copy. No extensions will be granted if you lose your work and you had no submit server backups.

Piazza

We will be using (Piazza) for class communication. You will not be able to register to Piazza yourself. Your instructor will register you using the email address you have in the school system. Posting of any kind of code in Piazza is not allowed.

Class Announcements

You are responsible for checking announcements (at least twice a day) we post in the announcements Piazza folder. An oldannouncements Piazza folder will have old announcements.

Excused Absence and Academic Accommodations

See the section titled "Attendance and Missed Assignments" available at Course Related Policies.

Even if you have a valid reason for missing a quiz or an exam, you still have to let Pedram or Nelson (not a TA) know BEFORE the time of the exam/quiz. We have no obligation to provide a makeup even with a valid excuse if you send us an email AFTER the exam/quiz. Simply put, if there is an issue, say something as soon as the issue comes up.

As for a note from a physician, please provide one from the UHC or a local office, where you can get an actual signature. Notes provided by an online service that never see you in person will not be acceptable.

Accessibility

See the section titled "Accessibility" available at Course Related Policies.

For Accessibility & Disability (ADS) Students

If you are an ADS (https://counseling.umd.edu/ads) student (others ignore this information):

ADS students: you are responsible for reserving a space at ADS to take exams (we cannot provide that support). Keep in mind ADS has deadlines regarding by when to schedule a day/time to take exams. If your main accommodation is extra time in exams and quizzes, you don't need to meet your instructor (just bring any form that needs a signature to lecture).

Academic Integrity

Please read this information carefully. We take academic integrity matters seriously.

1. Academic dishonesty includes not only cheating, fabrication, and plagiarism, but also includes helping other students commit acts of academic dishonesty by allowing them to obtain copies of your work. All submitted work must be your own. Cases of academic dishonesty will be pursued to the fullest extent possible as stipulated by the Office of Student Conduct.
2. Situations that often lead to academic integrity violations:
    - A student's friend/roommate shares an assignment's code. Once you provide your code to another student, you are a facilitator, even if you indicate to the student "not to copy-paste" any of it. Actually we had a case in which a student CS degree was revoked for this reason.
    - Students use online resources (github, Chegg, etc.) to find assignments' solutions. The solutions are found by several students and all will be involved in an academic case.

- Students assume we don't have tools that check for similarities among all students' submissions.
- Students get desperate and don't want a 0 in the assignment.
- Students are not aware of the expectations regarding academic integrity.
- Students assume we don't take academic integrity matters seriously.
- You should only receive assistance from instructors/TAs. We have seen cases in which the use of tutors have led to academic integrity violations (e.g., tutors looked for assignment's solutions online).

3. The Office of Student Conduct is responsible for handling academic integrity matters. After a report is submitted by an instructor, the case is evaluated by the office and it could result in an XF grade, degree revocation, or dismissal from the university.
4. One of the most negative consequences of academic integrity violations is the emotional burden an academic integrity case has on a student. We have seen students extremely distraught as a result of an academic integrity violation. In many cases students chances for recommendations, TA positions, and other opportunities are negatively affected.
5. Please read the section titled "Academic Integrity" available at Course Related Policies and the information available at Academic Integrity

Class Concerns

If you or your parents have any class concerns, feel free to contact the instructor. If an issue arises with the instructor, report it using the form available at https://www.cs.umd.edu/classconcern.

Notice of Mandatory Reporting

Notice of mandatory reporting of sexual assault, sexual harassment, interpersonal violence, and stalking: As faculty members, the course instructors are designated as a "Responsible University Employee," and we must report all disclosures of sexual assault, sexual harassment, interpersonal violence, and stalking to UMD's Title IX Coordinator per University Policy on Sexual Harassment and Other Sexual Misconduct.

If you wish to speak with someone confidentially, please contact one of UMD's confidential resources, such as CARE to Stop Violence (located on the Ground Floor of the Health Center) at 301-741-3442 or the Counseling Center (located at the Shoemaker Building) at 301-314-7651.

You may also seek assistance or supportive measures from UMD's Title IX Coordinator, Angela Nastase, by calling 301-405-1142, or emailing titleIXcoordinator@umd.edu.

To view further information on the above, please visit the Office of Civil Rights and Sexual Misconduct's website at ocrsm.umd.edu.

Miscellaneous

- We only use ELMS for videos.
- At the end of the semester visit (www.courseevalum.umd.edu) to complete your course evaluations.
- Contact your instructor, the Counseling Center , or both, if you are experiencing difficulties that affect your performance in your courses. Do not wait until the end of the semester to look for help.

- UMD Course related policies are available at http://www.ugst.umd.edu/courserelatedpolicies.html.

Copyright

Appendix D

Demographics Survey

This survey will ask you some questions about your demographics. Please read and answer questions to the best of your ability.

1. What is your name?
2. What is your UID?
3. What year are you?
    a. Freshman
    b. Sophomore
    c. Junior
    d. Senior
    e. Other_____
4. What is your major?
5. How old are you?
    a. 18
    b. 19
    c. 20
    d. 21
    e. 22
    f. 23
    g. Other_____
6. What is your gender identity?
    a. Man
    b. Woman
    c. Non-binary

   d. Agender
   e. Gender Fluid
   f. Intersex
   g. I would rather not answer
   h. Other_____

7. What is your ethnicity? (Check all that apply)
   a. Caucasian/White
   b. African-American/Black
   c. Latino/Hispanic
   d. Asian
   e. Native
   f. I would rather not answer
   g. Other_____

8. Are you a first generation college student?
   a. Yes
   b. No
   c. Not sure
   d. I would rather not answer

9. Is English your first language?
   a. Yes
   b. No
   c. Not sure
   d. I would rather not answer

10. On average, how many hours a week do you work at a job during the semester?
   a. 0-1
   b. 2-5

c. 6-10

d. 11-15

e. 16-20

f. More than 20 hours

11. Which of the following responsibilities do you have outside of school work? (Check all that apply)

a. Financial support for yourself

b. Financial support for family

c. Childcare children and/or siblings

d. Care for one or more older people

e. Cleaning living environment

f. Buying and/or cooking food

g. Care for an animal

h. Other_____

12. Did you learn programming in high school or extracurricular activities (e.g.,afterschool club, library group, summer camp)?

a. Yes

b. No

c. Not Sure

d. I would rather not answer

13. How many computer science courses have you taken at a college level (includingAP)?

a. 0

b. 1

c. 2

d. 3

e. 4

f. 5 or more

14. What programming languages have you worked with before taking this course?
15. Did an adult you know growing up work in a computer science related field?
    a. Yes
    b. No
    c. Not sure
    d. I would rather not answer
16. Were you in gifted or honors programs in high school or currently?
    a. Yes
    b. No
    c. Not sure
    d. I would rather not answer
17. Were you ever in special education programs?
    a. Yes
    b. No
    c. Not sure
    d. I would rather not answer
18. Did you ever have a private tutor?
    a. Yes
    b. No
    c. Not sure
    d. I would rather not answer
19. Are you registered with Accessibility & Disability Service (ADS) at UMD?
    a. Yes
    b. No
    c. Not sure

d. I would rather not answer

20. If you are registered with ADS at UMD, what accommodations have been approved for you to use?
21. If you are registered with ADS at UMD, what accommodations do you actually usein your classes?
22. Have you been formally diagnosed with any of the following?
    a. ADHD (Attention Deficit Hyperactivity Disorder)
    b. ASD (Autism Spectrum Disorder)
    c. Dyspraxia
    d. Dyslexia
    e. Dyscalculia
    f. Dysgraphia
    g. Tourette's Syndrome
    h. Bipolar Disorder
    i. Depressive Disorders
    j. Anxiety Disorders
    k. None of the above
    l. Other_____
23. Self-diagnosis refers to a person doing significant research to conclude that they might have a specific condition. Which, if any, of the following have you self-diagnosed?
    a. ADHD (Attention Deficit Hyperactivity Disorder)
    b. ASD (Autism Spectrum Disorder)
    c. Dyspraxia
    d. Dyslexia
    e. Dyscalculia

f. Dysgraphia

g. Tourette's Syndrome

h. Bipolar Disorder

i. Depressive Disorders

j. Anxiety Disorders

k. None of the above

l. Other_____

24. Do you have any chronic illnesses or other physical disabilities that affect your school work?

a. Yes

b. No

c. Not Sure

d. I would rather not answer

Appendix E

Personality Survey/RAADS-R

This survey will ask you some questions about your personality. Please read and answer questions to the best of your ability.

1. I am a sympathetic person.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
2. I often use words and phrases from movies and television in conversations.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
3. I am often surprised when others tell me I have been rude.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
4. Sometimes I talk too loudly or too softly, and I am not aware of it.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
5. I often don't know how to act in social situations.

a. True now and when I was young
b. True only now
c. True only when I was younger than 16
d. Never true

6. I can 'put myself in other people's shoes.'
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
7. I have a hard time figuring out what some phrases mean, like 'you are the apple of my eye.'
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
8. I only like to talk to people who share my special interests.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
9. I focus on details rather than the overall idea.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true

10. I always notice how food feels in my mouth. This is more important to me than how it tastes.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
11. I miss my best friends or family when we are apart for a long time.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
12. Sometimes I offend others by saying what I am thinking, even if I don't mean to.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
13. I only like to think and talk about a few things that interest me.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
14. I'd rather go out to eat in a restaurant by myself than with someone I know.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true

15. I cannot imagine what it would be like to be someone else.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
16. I have been told that I am clumsy or uncoordinated.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
17. Others consider me odd or different.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
18. I understand when friends need to be comforted.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
19. I am very sensitive to the way my clothes feel when I touch them. How they feel is more important to me than how they look.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true

20. I like to copy the way certain people speak and act. It helps me appear more normal.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
21. It can be very intimidating for me to talk to more than one person at the same time.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
22. I have to 'act normal' to please other people and make them like me.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
23. Meeting new people is usually easy for me.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
24. I get highly confused when someone interrupts me when I am talking about something I am very interested in.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true

25. It is difficult for me to understand how other people are feeling when we are talking.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
26. I like having a conversation with several people, for instance around a dinner table, at school or at work.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
27. I take things too literally, so I often miss what people are trying to say.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
28. It is very difficult for me to understand when someone is embarrassed or jealous.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
29. Some ordinary textures that do not bother others feel very offensive when they touch my skin.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16

d. Never true

30. I get extremely upset when the way I like to do things is suddenly changed.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
31. I have never wanted or needed to have what other people call an 'intimate relationship.'
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
32. It is difficult for me to start and stop a conversation. I need to keep going until I am finished.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
33. I speak with a normal rhythm.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
34. The same sound, color or texture can suddenly change from very sensitive to very dull.
    a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

35. The phrase 'I've got you under my skin' makes me uncomfortable.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

36. Sometimes the sound of a word or a high-pitched noise can be painful to my ears.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

37. I am an understanding type of person.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

38. I do not connect with characters in movies and cannot feel what they feel.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

39. I cannot tell when someone is flirting with me.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

40. I can see in my mind in exact detail things that I am interested in.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

41. I keep lists of things that interest me, even when they have no practical use (for example sports statistics, train schedules, calendar dates, historical facts and dates).

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

42. When I feel overwhelmed by my senses, I have to isolate myself to shut them down.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

43. I like to talk things over with my friends.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

44. I cannot tell if someone is interested or bored with what I am saying.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

45. It can be very hard to read someone's face, hand and body movements when they are talking.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

46. The same thing (like clothes or temperatures) can feel very different to me at different times.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

47. I feel very comfortable with dating or being in social situations with others.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

48. I try to be as helpful as I can when other people tell me their personal problems.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

49. I have been told that I have an unusual voice (for example flat, monotone, childish, or high-pitched).

a. True now and when I was young
b. True only now
c. True only when I was younger than 16
d. Never true

50. Sometimes a thought or a subject gets stuck in my mind and I have to talk about it even if no one is interested.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
51. I do certain things with my hands over and over again (like flapping, twirling sticks or strings, waving things by my eyes).
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
52. I have never been interested in what most of the people I know consider interesting.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
53. I am considered a compassionate type of person.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true

54. I get along with other people by following a set of specific rules that help me look normal.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
55. It is very difficult for me to work and function in groups.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
56. When I am talking to someone, it is hard to change the subject. If the other person does so, I can get very upset and confused.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
57. Sometimes I have to cover my ears to block out painful noises (like vacuum cleaners or people talking too much or too loudly).
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
58. I can chat and make small talk with people.
    a. True now and when I was young
    b. True only now

c. True only when I was younger than 16

d. Never true

59. Sometimes things that should feel painful are not (for instance when I hurt myself or burn my hand on the stove).

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

60. When talking to someone, I have a hard time telling when it is my turn to talk or to listen.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

61. I am considered a loner by those who know me best.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

62. I usually speak in a normal tone.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

63. I like things to be exactly the same day after day and even small changes in my routines upset me.

a. True now and when I was young
b. True only now
c. True only when I was younger than 16
d. Never true

64. How to make friends and socialize is a mystery to me.
a. True now and when I was young
b. True only now
c. True only when I was younger than 16
d. Never true

65. It calms me to spin around or to rock in a chair when I'm feeling stressed.
a. True now and when I was young
b. True only now
c. True only when I was younger than 16
d. Never true

66. The phrase, 'He wears his heart on his sleeve,' does not make sense to me.
a. True now and when I was young
b. True only now
c. True only when I was younger than 16
d. Never true

67. If I am in a place where there are many smells, textures to feel, noises or bright lights, I feel anxious or frightened.
a. True now and when I was young
b. True only now
c. True only when I was younger than 16
d. Never true

68. I can tell when someone says one thing but means something else.

a. True now and when I was young
b. True only now
c. True only when I was younger than 16
d. Never true

69. I like to be by myself as much as I can.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
70. I keep my thoughts stacked in my memory like they are on filing cards, and I pick out the ones I need by looking through the stack and finding the right one (or another unique way).
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
71. The same sound sometimes seems very loud or very soft, even though I know it has not changed.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16
    d. Never true
72. I enjoy spending time eating and talking with my family and friends.
    a. True now and when I was young
    b. True only now
    c. True only when I was younger than 16

d. Never true

73. I can't tolerate things I dislike (like smells, textures, sounds or colors).

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

74. I don't like to be hugged or held.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

75. When I go somewhere, I have to follow a familiar route or I can get very confused and upset.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

76. It is difficult to figure out what other people expect of me.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

77. I like to have close friends.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

78. People tell me that I give too much detail.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

79. I am often told that I ask embarrassing questions.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

80. I tend to point out other people's mistakes.

a. True now and when I was young

b. True only now

c. True only when I was younger than 16

d. Never true

Appendix F

After Exam Reflection Survey (Fall 2022)

This survey will ask you some questions about the class. Please read and answer questions to the best of your ability.

1. What is your name?
2. What is your UID?

Course Material

3. I find the course website to be helpful.

   Strongly Disagree 1 2 3 4 5 Strongly Agree
4. I find the slides to be helpful.

   Strongly Disagree 1 2 3 4 5 Strongly Agree
5. I find the sample code to be helpful.

   Strongly Disagree 1 2 3 4 5 Strongly Agree
6. What type of course material do you find is the most helpful? Why?
7. What type of course material do you find is the least helpful? Why?
8. I understand the topic intro to computer systems.

   Strongly Disagree 1 2 3 4 5 Strongly Agree
9. I understand the topic variables and operators.

   Strongly Disagree 1 2 3 4 5 Strongly Agree
10. I understand the topic expressions, statements, and methods.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

11. I understand the topic Java text and input/output.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

12. I understand the topic conditionals.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

13. I understand the topic loops.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

14. I understand the topic principles of object oriented programming.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

15. I understand the topic basics of program design.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

16. I understand the topic testing and debugging.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

17. I understand the topic Java memory map.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

18. I understand the topic arrays and Java arraylists.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

19. I understand the topic Java interfaces.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

20. I understand the topic inheritance overview.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

21. I understand the topic recursion.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

Programming Assignments (Projects)

22. The programming assignments help me to understand the material we learn in class.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

23. I feel I have enough time to complete the programming assignments.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

24. Do you feel your programming assignment code reflects your understanding of the material? Why?

Quizzes and Exams

25. I feel that the quizzes accurately show how much I understand the material.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

26. I feel that I have enough time to complete quizzes.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

27. Do you feel prepared for the quizzes?

28. I feel that the exams accurately show how much I understand the material.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

29. I feel that I have enough time to complete exams.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

30. Do you feel prepared for the exams?

Personal Questions

31. I go to class often.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

32. I go to discussions often.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

33. How many hours a week have you spent in office hours with the professor?

a. 0 to 30 mins

b. 30 mins to 1 hour

c. 1 to 1.5 hours

d. 1.5 to 2 hours

e. 2 to 2.5 hours

f. 2.5 to 3 hours

34. I find that office hours with the professor are very helpful.

Strongly Disagree 1 2 3 4 5 Strongly Agree

35. How many hours a week have you spent in office hours with the TAs?

a. Less than 1 hour

b. 1 to 2 hours

c. 2 to 3 hours

d. 3 to 4 hours

e. 4 to 5 hours

f. 5 or more hour

36. I find that office hours with the TA's are very helpful.

Strongly Disagree 1 2 3 4 5 Strongly Agree

37. This class is my hardest class.

Strongly Disagree 1 2 3 4 5 Strongly Agree

38. This class is stressful.

Strongly Disagree 1 2 3 4 5 Strongly Agree

39. This class has negatively effected my mental health.

Strongly Disagree 1 2 3 4 5 Strongly Agree

40. I feel like I have enough time to dedicate to this class.

Strongly Disagree 1 2 3 4 5 Strongly Agree

41. I feel like I will pass this course with a grade I am satisfied with.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

42. I like the material of the course.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

43. Is there anything that would help you that was not provided to you?
44. Is there anything else that you would want people to know about how the course is structured?

After Exam Reflection Survey (Fall 2023)

This survey will ask you some questions about the class. Please read and answer questions to the best of your ability.

1. What is your name?
2. What is your UID?

Course Material

3. I find the course website to be helpful.

   Strongly Disagree 1 2 3 4 5 Strongly Agree

4. I find the slides to be helpful.

   Strongly Disagree 1 2 3 4 5 Strongly Agree

5. I find the sample code to be helpful.

   Strongly Disagree 1 2 3 4 5 Strongly Agree

6. What type of course material do you find is the most helpful? Why?
7. What type of course material do you find is the least helpful? Why?
8. I understand the topic intro to computer systems.

   Strongly Disagree 1 2 3 4 5 Strongly Agree

9. I understand the topic variables and operators.

   Strongly Disagree 1 2 3 4 5 Strongly Agree

10. I understand the topic expressions, statements, and methods.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

11. I understand the topic Java text and input/output.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

12. I understand the topic conditionals.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

13. I understand the topic loops.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

14. I understand the topic principles of object oriented programming.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

15. I understand the topic basics of program design.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

16. I understand the topic testing and debugging.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

17. I understand the topic Java memory map.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

18. I understand the topic arrays and Java arraylists.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

19. I understand the topic Java interfaces.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

20. I understand the topic inheritance overview.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

21. I understand the topic recursion.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

Programming Assignments (Projects)

22. The programming assignments help me to understand the material we learn in class.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

23. I feel I have enough time to complete the programming assignments.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

24. Do you feel your programming assignment code reflects your understanding of the material? Why?
25. Do you feel the programming assignments written descriptions are clear? Why?
26. Is there anything we could provide you that would help you understand the programming assignments more?

Quizzes and Exams

27. I feel that the quizzes accurately show how much I understand the material.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

28. I feel that I have enough time to complete quizzes.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

29. Do you feel prepared for the quizzes?
30. I feel that the exams accurately show how much I understand the material.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

31. I feel that I have enough time to complete exams.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

32. Do you feel prepared for the exams?

Personal Questions

33. I go to class often.

Strongly Disagree 1 2 3 4 5 Strongly Agree

34. I go to discussions often.

Strongly Disagree 1 2 3 4 5 Strongly Agree

35. How many hours a week have you spent in office hours with the professor?

a. 0 to 30 mins

b. 30 mins to 1 hour

c. 1 to 1.5 hours

d. 1.5 to 2 hours

e. 2 to 2.5 hours

f. 2.5 to 3 hours

36. I find that office hours with the professor are very helpful.

Strongly Disagree 1 2 3 4 5 Strongly Agree

37. How many hours a week have you spent in office hours with the TAs?

a. Less than 1 hour

b. 1 to 2 hours

c. 2 to 3 hours

d. 3 to 4 hours

e. 4 to 5 hours

f. 5 or more hour

38. I find that office hours with the TA's are very helpful.

Strongly Disagree 1 2 3 4 5 Strongly Agree

39. This class is my hardest class.

Strongly Disagree 1 2 3 4 5 Strongly Agree

40. This class is stressful.

Strongly Disagree 1 2 3 4 5 Strongly Agree

41. This class has negatively effected my mental health.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

42. I feel like I have enough time to dedicate to this class.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

43. I feel like I will pass this course with a grade I am satisfied with.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

44. I like the material of the course.

    Strongly Disagree 1 2 3 4 5 Strongly Agree

45. Is there anything that would help you that was not provided to you?

46. Is there anything else that you would want people to know about how the course is structured?

Appendix G

For Nelson

1. How long have you been teaching introductory programming courses?
2. How have you worked with Accessibility and Disability Services students in the past?
    a. Generally, do you read the letter, go off of what the students say, etc?
3. Have students disclosed any disabilities to you in the past?
    a. What types of disabilities have you worked with?
    b. How have you accommodated these students beyond their letters/ even if they aren't registered with Accessibility and Disability Services?
4. If you could change something about the course without any constraints, what would it be?
5. Do you think that students that struggle in introductory programming courses are prepared for the course when they enter college?
    a. Why do you think so?
    b. How would you help these students?
6. In general, what concepts do students usually struggle with?
7. Introductory programming courses have a high dropout and failure rate, why do you think students struggle to learn programming?
8. Do you think that there is a type of student that will never struggle no matter how challenging the material is? Describe them.
9. Based on past experience, how might you help a student who is struggling with communication (lecture changes, exam changes, policy changes)?
10. Based on past experience, how might you help a student who is struggling with talking to and working with peers?
11. Based on past experience, how might you help a student who is struggling with learning and comprehending arithmetic?
12. Based on past experience, how might you help a student who is struggling with attention?
13. Based on past experience, how might you help a student who is struggling with coming

to class?

14. Based on past experience, how might you help a student who is struggling with processing language?
15. Based on past experience, how might you help a student who is struggling with mental and emotional health issues like depression, anxiety, self-confidence, and stress?

For the TA's Spring 2023

1. What is your university status (undergraduate/ graduate student and what year)?
2. Have you TA'ed any other introductory programming courses?
3. What are your responsibilities as a TA (grade, hold office hours, teach discussion, ect..)?
4. How have you worked with Accessibility and Disability Services students in the past?

Do you have any experience with IEPs?

Do you have any personal experience with people who are neurodiverse out side of school

5. Have students disclosed any disabilities to you in the past?
    a. What types of disabilities have you worked with?
    b. How are you notified that a student has a disability?
    c. How would you accommodate these students beyond their letters/ even if they aren't registered with Accessibility and Disability Services?
6. Introductory programming courses have a high dropout and failure rate, why do you think students struggle to learn programming?
7. Based on past experience, how might you help a student who is struggling with communication (lecture changes, exam changes, policy changes)?
8. Based on past experience, how might you help a student who is struggling with talking to and working with peers?
9. Based on past experience, how might you help a student who is struggling with learning and comprehending arithmetic?
10. Based on past experience, how might you help a student who is struggling with attention?
11. Based on past experience, how might you help a student who is struggling with coming to class?

12. Based on past experience, how might you help a student who is struggling with processing language?
13. Based on past experience, how might you help a student who is struggling with mental and emotional health issues like depression, anxiety, self-confidence, and stress?

Are there times that you have also experience what students struggle

For the TA's Fall 2023

14. What is your university status (undergraduate/ graduate student and what year)?
15. Have you TA'ed any other introductory programming courses?
16. How have you worked with Accessibility and Disability Services students in the past?
17. Have students disclosed any disabilities to you in the past?
    a. What types of disabilities have you worked with
    b. How would you accommodate these students beyond their letters/ even if they aren't registered with Accessibility and Disability Services?
18. What are your responsibilities as a TA (grade, hold office hours, teach discussion, ect..)?
19. Do a lot of students come to office hours?
20. What are some typical questions that students would come to you with during office hours?
21. How would you help a learning disabled student in office hours?
22. How would you help a student who struggles with their mental and emotional health in office hours?
23. What did the discussion look like?
    a. Did many students come?
    b. Were the students active participants during your discussion?
    c. Was there anything that you wish you could have done differently in discussion?
    d. How would you help a student in discussion that has a learning disability?
    e. How would you help a student in discussion that struggles with their mental and emotional health?
24. How well do you think students understood the concepts when they were asking

questions?

   a. Are there any concepts that students seem to really struggle with?

25. Did you use the course website resources at all?
    a. If so, how?
    b. Did you find them useful?
26. Did the students complain about anything during the semester?
27. Do you feel that the semester went well?
    a. What does the semester going well mean to you?

For the Students Spring 2023

1. Did you have Nelson for CMSC 131?
2. How do you feel about CMSC 131?
    a. Did you enjoy it?
    b. Did you find the instructor taught well?
        i. What does teaching well mean to you?
    c. Do you find that the TA leads discussion well?
        i. What does leading discussion well mean to you?
    d. Do you feel like you learned a lot from the class? Why or Why not?
    e. Do you feel like this course prepared you for programming in the real world? Why or why not?
    f. Was there anything that might have affected your performance in this course?
    g. What stressed you out the most about this course? Why?
    h. How was your mental health during the semester you took CMSC 131?
        i. What were some things that you did to help your mental health during the semester?
        ii. Was there anything that your instructor could have done to help your mental health?
        iii. Was there anything that any of your TA's could have done to help your

mental health?

3. How do you feel about CMSC 132?
    a. Did you enjoy it?
    b. Did you find the instructor taught well?
        i. What does teaching well mean to you?
    c. Do you find that the TA leads discussion well?
        i. What does leading discussion well mean to you?
    d. Do you feel like you learned a lot from the class? Why or Why not?
    e. Do you feel like this course prepared you for programming in the real world? Why or why not?
    f. Was there anything that might have affected your performance in this course?
    g. What stressed you out the most about this course? Why?
    h. How was your mental health during the semester you took CMSC 132?
        i. What were some things that you did to help your mental health during the semester?
        ii. Was there anything that your instructor could have done to help your mental health?
        iii. Was there anything that any of your TA's could have done to help your mental health?
    i. How often did you use the resources on the course website?
        i. What did you use the most?
        ii. What did you use the least? Why?
    j. How did you feel about looking at code while learning the material?
        i. Were you able to follow along?
        ii. Did you ever zone out while watching them?
        iii. Do you find you understand the sample code after a looking at the code in class?
    k. Was there anything that could have been done differently to the lectures that

would have helped you during the semester?

l. Was there anything that could have been done differently to the programming assignments that would have helped you during the semester?

m. Was there anything that could have been done differently to the exams that would have helped you during the semester?

n. Are there any resources that would have been helpful that were not provided for you?

4. Do you feel like you belong in the CS program here? Why or why not?

Has there been any instances where other students made you feel like you don't belong?

For the Students Fall 2023

1. How do you feel about CMSC 131?
    a. Did you enjoy it?
    b. Did you find the instructor taught well?
        i. What does teaching well mean to you?
    c. Do you find that the TA leads discussion well?
        i. What does leading discussion well mean to you?
    d. Do you feel like you learned a lot from the class? Why or Why not?
    e. Do you feel like this course prepared you for programming in the real world? Why or why not?
    f. Was there anything that might have affected your performance in this course?
    g. What stressed you out the most about this course? Why?
    h. How was your mental health during the semester you took CMSC 131?
        i. What were some things that you did to help your mental health during the semester?
        ii. Was there anything that your instructor could have done to help your mental health?
        iii. Was there anything that any of your TA's could have done to help your mental health?

i. How often did you use the resources on the course website?
    i. What did you use the most?
    ii. What did you use the least? Why?
j. Did you notice anything different about project 5 compared to the rest of the projects?
k. How often did you use the Animations and Images to help you understand concepts?
l. How did you feel about the live coding sessions?
    i. Were you able to follow along?
    ii. Did you ever zone out while watching them?
    iii. Do you find you understand the sample code after a live coding session?
m. How do you feel about coding on paper for an exam?
n. Did you like being able to code in an IDE as part of the exam? Why?
o. Was there anything that could have been done differently to the lectures that would have helped you during the semester?
p. Was there anything that could have been done differently to the programming assignments that would have helped you during the semester?
q. Was there anything that could have been done differently to the exams that would have helped you during the semester?
r. Are there any resources that would have been helpful that were not provided for you?

2. Do you feel like you belong in the CS program here? Why or why not?

Bibliography

Abascal, J. (2002). Human-computer interaction in assistive technology: From "Patchwork" to "Universal Design." *IEEE International Conference on Systems, Man and Cybernetics*, *3*, 6 pp. vol.3-. https://doi.org/10.1109/ICSMC.2002.1176076

ADA National Network. (2024, February). *What are a public or private college-university's responsibilities to students with disabilities? | ADA National Network*. Information, Guidance, and Training on the Americans with Disability Act. https://adata.org/faq/what-are-public-or-private-college-universitys-responsibilities-students-disabilities

Advokat, C. D., & Scheithauer, M. (2013). Attention-deficit hyperactivity disorder (ADHD) stimulant medications as cognitive enhancers. *Frontiers in Neuroscience*, *7*. https://doi.org/10.3389/fnins.2013.00082

Aguar, K., Arabnia, H. R., Gutierrez, J. B., Potter, W. D., & Taha, T. R. (2016). Making CS Inclusive: An Overview of Efforts to Expand and Diversify CS Education. *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, 321–326. https://doi.org/10.1109/CSCI.2016.0067

Akoumianakis, D., & Stephanidis, C. (2001). Universal Design in HCI: A critical review of current research and practice. *Undefined*. https://www.semanticscholar.org/paper/Universal-Design-in-HCI%3A-A-critical-review-of-and-Akoumianakis-Stephanidis/1355a53c22313330fcadc393598414ad04e6e029

Ambrósio, A. P., Costa, F. M., Almeida, L., Franco, A., & Macedo, J. (2011). Identifying cognitive abilities to improve CS1 outcome. *2011 Frontiers in Education Conference (FIE)*, F3G-1-F3G-7. https://doi.org/10.1109/FIE.2011.6142824

American Psychiatric Association. (2022). *The Diagnostic and Statistical Manual of Mental*

Disorders (5th Edition Text Revision). American Psychiatric Association.

Anderson, A. H., Carter, M., & Stephenson, J. (2018). Perspectives of University Students with Autism Spectrum Disorder. *Journal of Autism and Developmental Disorders*, *48*(3), 651–665. https://doi.org/10.1007/s10803-017-3257-3

Asbell-Clarke, J., Rowe, E., Almeda, V., Edwards, T., Bardar, E., Gasca, S., Baker, R. S., & Scruggs, R. (2021). The development of students' computational thinking practices in elementary- and middle-school classes using the learning game, Zoombinis. *Computers in Human Behavior*, *115*, 106587. https://doi.org/10.1016/j.chb.2020.106587

Baglama, B., Yücesoy, Y., & Yıkmış, A. (2018). Using Animation as a Means of Enhancing Learning of Individuals with Special Needs. *TEM Journal*, *7*, 670–677. https://doi.org/10.18421/TEM73-26

Barnhill, G. P. (2016). Supporting Students With Asperger Syndrome on College Campuses: Current Practices. *Focus on Autism and Other Developmental Disabilities*, *31*(1), 3–15. https://doi.org/10.1177/1088357614523121

Beaubouef, T., & Mason, J. (2005). Why the high attrition rate for computer science students: Some thoughts and observations. *ACM SIGCSE Bulletin*, *37*(2), 103–106. https://doi.org/10.1145/1083431.1083474

Bhujade, V. M. (2017). Depression, anxiety and academic stress among college students: A brief review. *Indian Journal of Health and Wellbeing*, *8*(7), 748–751.

Birdwell, M. L. N., & Bayley, K. (2022). When the Syllabus Is Ableist: Understanding How Class Policies Fail Disabled Students. *Teaching English in the Two Year College*, *49*(3), 220–237.

Boat, T. F., Wu, J. T., Disorders, C. to E. the S. S. I. D. P. for C. with M., Populations, B. on the H. of S., Board on Children, Y., Medicine, I. of, Education, D. of B. and S. S. and, & The National Academies of Sciences, E. (2015). Prevalence of Learning Disabilities.

In *Mental Disorders and Disabilities Among Low-Income Children*. National Academies Press (US). https://www.ncbi.nlm.nih.gov/books/NBK332880/

Bolourian, Y., Zeedyk, S. M., & Blacher, J. (2018). Autism and the University Experience: Narratives from Students with Neurodevelopmental Disorders. *Journal of Autism and Developmental Disorders*, *48*(10), 3330–3343. https://doi.org/10.1007/s10803-018-3599-5

Bone, E. K., & Bouck, E. C. (2017). Accessible text-to-speech options for students who struggle with reading. *Preventing School Failure: Alternative Education for Children and Youth*, *61*(1), 48–55. https://doi.org/10.1080/1045988X.2016.1188366

Borsotti, V., Begel, A., & Bjørn, P. (2024). Neurodiversity and the Accessible University: Exploring Organizational Barriers, Access Labor and Opportunities for Change. *Proceedings of the ACM on Human-Computer Interaction*, *8*(CSCW1), 172:1-172:27. https://doi.org/10.1145/3641011

Boutemen, L., & Miller, A. N. (2023). Readability of publicly available mental health information: A systematic review. *Patient Education and Counseling*, *111*, 107682. https://doi.org/10.1016/j.pec.2023.107682

Bryne, M., Catrambone, R., & Stasko, J. T. (1999). Evaluating animations as student aids in learning computer algorithms. *Computers & Education*, *33*(4), 253–278. https://doi.org/10.1016/S0360-1315(99)00023-8

CDC. (2020a, March 26). *Autism and Developmental Disabilities Monitoring (ADDM) Network | CDC*. Centers for Disease Control and Prevention. https://www.cdc.gov/ncbddd/autism/addm.html

CDC. (2020b, November 16). *Data and Statistics About ADHD | CDC*. Centers for Disease Control and Prevention. https://www.cdc.gov/ncbddd/adhd/data.html

CDC. (2021, June 1). *Data and Statistics on Tourette Syndrome | CDC*. Centers for Disease Control and Prevention. https://www.cdc.gov/ncbddd/tourette/data.html

Chrysochoou, M., Zaghi, A. E., & Syharat, C. M. (2022). Reframing neurodiversity in engineering education. *Frontiers in Education*, *7*. https://doi.org/10.3389/feduc.2022.995865

Cipolla, C. (2018). "Not so Backwards": A Phenomenological Study on the Lived Experiences of High Achieving Post-Secondary Students with Dyslexia. In *ProQuest LLC*. ProQuest LLC.

Clouder, L., Karakus, M., Cinotti, A., Ferreyra, M. V., Fierros, G. A., & Rojo, P. (2020). Neurodiversity in higher education: A narrative synthesis. *Higher Education*, *80*(4), 757–778. https://doi.org/10.1007/s10734-020-00513-6

Cohen, R. F., Fairley, A. V., Gerry, D., & Lima, G. R. (2005). Accessibility in introductory computer science. *ACM SIGCSE Bulletin*, *37*(1), 17–21. https://doi.org/10.1145/1047124.1047367

Couzens, D., Poed, S., Kataoka, M., Brandon, A., Hartley, J., & Keen, D. (2015). Support for Students with Hidden Disabilities in Universities: A Case Study. *International Journal of Disability, Development and Education*, *62*(1), 24–41. https://doi.org/10.1080/1034912X.2014.984592

Demer, L. L. (2018). The Autism Spectrum: Human Rights Perspectives. *Pediatrics*, *141*(Supplement_4), S369–S372. https://doi.org/10.1542/peds.2016-4300O

Doyle, N. (2020). Neurodiversity at work: A biopsychosocial model and the impact on working adults. *British Medical Bulletin*, *135*(1), 108–125. https://doi.org/10.1093/bmb/ldaa021

Draaisma, D. (2009). Stereotypes of autism. *Philosophical Transactions of the Royal Society B*, *364*(1522). https://doi-org.proxy-um.researchport.umd.edu/10.1098/rstb.2008.0324

Dunn, C., Rabren, K. S., Taylor, S. L., & Dotson, C. K. (2012). Assisting Students With High-Incidence Disabilities to Pursue Careers in Science, Technology, Engineering,

and Mathematics. *Intervention in School and Clinic*, *48*(1), 47–54. https://doi.org/10.1177/1053451212443151

DuPaul, G. J., Dahlstrom-Hakki, I., Gormley, M. J., Fu, Q., Pinho, T. D., & Banerjee, M. (2017). College Students With ADHD and LD: Effects of Support Services on Academic Performance. *Learning Disabilities Research & Practice*, *32*(4), 246–256. https://doi.org/10.1111/ldrp.12143

Dwyer, P. (2022). The Neurodiversity Approach(es): What Are They and What Do They Mean for Researchers? *Human Development*, *66*(2), 73–92. https://doi.org/10.1159/000523723

Engelbrecht, & Silvertant, E. (2020, April 4). *RAADS–R | Embrace Autism*. https://embrace-autism.com/raads-r/

Fabio, R., & Antonietti, A. (2014). Hypermedia Tools Enhance Learning in ADHD Students. *ADHD Report The*, *6*, 8–9. https://doi.org/10.1521/adhd.2014.22.4.8

Fellowes, S. (2023). Self-Diagnosis in Psychiatry and the Distribution of Social Resources. *Royal Institute of Philosophy Supplements*, *94*, 55–76. https://doi.org/10.1017/S1358246123000218

Fuertes, J. L., Gonzalez, L. F., & Martinez, L. (2018). Visual Programming Languages for Programmers with Dyslexia: An Experiment. *2018 IEEE 14th International Conference on E-Science (e-Science)*, 145–155. https://doi.org/10.1109/eScience.2018.00030

Galeos, C., Karpouzis, K., & Tsatiris, G. (2020). Developing an educational programming game for children with ADHD. *2020 15th International Workshop on Semantic and Social Media Adaptation and Personalization (SMA*, 1–6. https://doi.org/10.1109/SMAP49528.2020.9248458

Get your document's readability and level statistics—Microsoft Support. (n.d.). Retrieved July 19, 2024, from https://support.microsoft.com/en-us/office/get-your-document-

s-readability-and-level-statistics-85b4969e-e80a-4777-8dd3-f7fc3c8b3fd2#ID0EBDD=Older_versions_of_Office

Getzel, E. E. (2008). Addressing the Persistence and Retention of Students with Disabilities in Higher Education: Incorporating Key Strategies and Supports on Campus. *Exceptionality*, *16*(4), 207–219. https://doi.org/10.1080/09362830802412216

Gibbs, J., Appleton, J., & Appleton, R. (2007). Dyspraxia or developmental coordination disorder? Unravelling the enigma. *Archives of Disease in Childhood*, *92*(6), 534–539. https://doi.org/10.1136/adc.2005.088054

Glazzard, J., & Dale, K. (2015). 'It Takes Me Half a Bottle of Whisky to Get through One of Your Assignments': Exploring One Teacher Educator's Personal Experiences of Dyslexia. *Dyslexia*, *21*(2), 177–192. https://doi.org/10.1002/dys.1493

Gobbo, K., & Shmulsky, S. (2020). Should Neurodiversity Culture Influence How Instructors Teach? *Academic Exchange Quarterly*, *23*.

Goering, S. (2015). Rethinking disability: The social model of disability and chronic disease. *Current Reviews in Musculoskeletal Medicine*, *8*(2), 134–138. https://doi.org/10.1007/s12178-015-9273-z

Green, A. L., & Rabiner, D. L. (2012). What Do We Really Know about ADHD in College Students? *Neurotherapeutics*, *9*(3), 559–568. https://doi.org/10.1007/s13311-012-0127-8

Griffin, E., & Pollak, D. (2009). Student experiences of neurodiversity in higher education: Insights from the BRAINHE project. *Dyslexia*, *15*(1), 23–41. https://doi.org/10.1002/dys.383

Habib, L., Berget, G., Sandnes, F. E., Sanderson, N., Kahn, P., Fagernes, S., & Olcay, A. (2012). Dyslexic students in higher education and virtual learning environments: An exploratory study. *Journal of Computer Assisted Learning*, *28*(6), 574–584. https://doi.org/10.1111/j.1365-2729.2012.00486.x

Hadjikakou, K., & Hartas, D. (2008). Higher education provision for students with disabilities in Cyprus. *Higher Education*, *55*(1), 103–119. https://doi.org/10.1007/s10734-007-9070-8

Hadley, W. (2017). The four-year college experience of one student with multiple learning disabilities. *College Student Journal*, *51*(1), 19–28.

Hadley, W. M. (2007). The Necessity of Academic Accommodations for First-Year College Students with Learning Disabilities. *Journal of College Admission*. https://eric.ed.gov/?id=EJ783943

Hamilton, L. G., & Petty, S. (2023). Compassionate pedagogy for neurodiversity in higher education: A conceptual analysis. *Frontiers in Psychology*, *14*. https://doi.org/10.3389/fpsyg.2023.1093290

Hanif, S., Fatima, W., Saeed, B., Khan, A., & Author, C. (2024). *AN ANALYSIS OF NATIONAL IDEOLOGICAL CONSTRUCTS IN EVOLVING PERSPECTIVES ON DISABILITY IN BUILT ENVIRONMENT DESIGN: A COMPREHENSIVE REVIEW OF THE SOCIAL AND MEDICAL MODELS IN CONTEMPORARY LITERATURE*.

Hansen, A. K., Hansen, E. R., Dwyer, H. A., Harlow, D. B., & Franklin, D. (2016). Differentiating for Diversity: Using Universal Design for Learning in Elementary Computer Science Education. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 376–381. https://doi.org/10.1145/2839509.2844570

Haynes-Magyar, C. (2024). Neurodiverse Programmers and the Accessibility of Parsons Problems: An Exploratory Multiple-Case Study. *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, 491–497. https://doi.org/10.1145/3626252.3630898

Hillier, A., Goldstein, J., Murphy, D., Trietsch, R., Keeves, J., Mendes, E., & Queenan, A. (2018). Supporting university students with autism spectrum disorder. *Autism*, *22*(1), 20–28. https://doi.org/10.1177/1362361317699584

Hsiao, F., Zeiser, S., Nuss, D., & Hatschek, K. (2018). Developing effective academic accommodations in higher education: A collaborative decision-making process. *International Journal of Music Education*, *36*(2), 244–258. https://doi.org/10.1177/0255761417729545

Initiative (WAI), W. W. A. (2024a, May 28). *Use White Spacing*. Web Accessibility Initiative (WAI). https://www.w3.org/WAI/WCAG2/supplemental/patterns/o3p10-whitespace/

Initiative (WAI), W. W. A. (2024b, May 29). *Ensure Foreground Content is not Obscured by Background*. Web Accessibility Initiative (WAI). https://www.w3.org/WAI/WCAG2/supplemental/patterns/o3p11-unobscured-foreground/

Initiative (WAI), W. W. A. (2024c, May 29). *Keep Text Succinct*. Web Accessibility Initiative (WAI). https://www.w3.org/WAI/WCAG2/supplemental/patterns/o3p05-succinct-text/

Initiative (WAI), W. W. A. (2024d, May 29). *Use a Consistent Visual Design*. Web Accessibility Initiative (WAI). https://www.w3.org/WAI/WCAG2/supplemental/patterns/o1p03-consistent-design/

Initiative (WAI), W. W. A. (2024e, May 29). *Use Clear Words*. Web Accessibility Initiative (WAI). https://www.w3.org/WAI/WCAG2/supplemental/patterns/o3p01-clear-words/

Israel, M., Jeong, G., Ray, M., & Lash, T. (2020). Teaching Elementary Computer Science through Universal Design for Learning. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 1220–1226). Association for Computing Machinery. http://doi.org/10.1145/3328778.3366823

Israel, M., Ray, M. J., Maa, W. C., Jeong, G. K., Lee, C. eun, Lash, T., & Do, V. (2018). School-embedded and district-wide instructional coaching in K-8 computer science: Implications for including students with disabilities. *Journal of Technology and Teacher Education*, *26*(3), 471–501.

Jones, A. M., West, K. B., & Suveg, C. (2019). Anxiety in the School Setting: A Framework for Evidence-Based Practice. *School Mental Health*, *11*(1), 4–14. https://doi.org/10.1007/s12310-017-9235-2

Ketterlin-Geller, L. R., & Johnstone, C. (2006). *Accommodations and Universal Design: Supporting Access to Assessments in Higher Education*. *19*(2), 10.

Killu, K., Marc, R., & Crundwell, A. (2016). Students with Anxiety in the Classroom: Educational Accommodations and Interventions. *Beyond Behavior*, *25*(2), 30–40. https://doi.org/10.1177/107429561602500205

Koushik, V., & Kane, S. K. (2019). "It Broadens My Mind": Empowering People with Cognitive Disabilities through Computing Education. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–12. https://doi.org/10.1145/3290605.3300744

Kreiser, N. L., & White, S. W. (2014). ASD in Females: Are We Overstating the Gender Difference in Diagnosis? *Clinical Child and Family Psychology Review*, *17*(1), 67–84. https://doi.org/10.1007/s10567-013-0148-9

Kurniawan, O., Lee, N. T. S., & Poskitt, C. M. (2020). Securing Bring-Your-Own-Device (BYOD) Programming Exams. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 880–886. https://doi.org/10.1145/3328778.3366907

Kuyath, S. (2002). *How Computer Animations Make Teaching Complex Topics More Effective And More Efficient*. 7.613.1-7.613.9. https://doi.org/10.18260/1-2--10240

Ladner, R. E., & Israel, M. (2016). "For all" in "computer science for all." *Communications of the ACM*, *59*(9), 26–28. https://doi.org/10.1145/2971329

Learning to write and writing to learn: Integrating communication skills into the computing curriculum | IEEE Conference Publication | IEEE Xplore. (n.d.). Retrieved July 13, 2024, from https://ieeexplore-ieee-org.proxy-um.researchport.umd.edu/abstract/document/475352

Lee, S. Y., & Mesfin, F. B. (2024). Blindness. In *StatPearls*. StatPearls Publishing. http://www.ncbi.nlm.nih.gov/books/NBK448182/

Make your PowerPoint presentations accessible to people with disabilities—Microsoft Support. (n.d.). Retrieved July 8, 2024, from https://support.microsoft.com/en-us/office/make-your-powerpoint-presentations-accessible-to-people-with-disabilities-6f7772b2-2f33-4bd2-8ca7-dae3b2b3ef25

Mallipeddi, N. V., & VanDaalen, R. A. (2021). Intersectionality Within Critical Autism Studies: A Narrative Review. *Autism in Adulthood*. https://doi.org/10.1089/aut.2021.0014

Mcgee, M. (2012). Neurodiversity. *Contexts*, *11*(3), 12–13. https://doi.org/10.1177/1536504212456175

McGrath, M. B., & Brown, J. R. (2005). Visual learning for science and engineering. *IEEE Computer Graphics and Applications*, *25*(5), 56–63. IEEE Computer Graphics and Applications. https://doi.org/10.1109/MCG.2005.117

Meaux, J. B., Green, A., & Broussard, L. (2009). ADHD in the college student: A block in the road. *Journal of Psychiatric and Mental Health Nursing*, *16*(3), 248–256. https://doi.org/10.1111/j.1365-2850.2008.01349.x

Medeiros, R. P., Ramalho, G. L., & Falcão, T. P. (2019). A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education. *IEEE Transactions on Education*, *62*(2), 77–90. IEEE Transactions on Education. https://doi.org/10.1109/TE.2018.2864133

Mehmood, E., Abid, A., Farooq, M. S., & Nawaz, N. A. (2020). Curriculum, Teaching and Learning, and Assessments for Introductory Programming Course. *IEEE Access*, *8*, 125961–125981. IEEE Access. https://doi.org/10.1109/ACCESS.2020.3008321

Mulholland, R., Pete, A. M., & Popeson, J. (2008). *Using Animated Language Software with Children Diagnosed with Autism Spectrum Disorders*. 9.

Munoz, R., Villarroel, R., Barcelos, T. S., Riquelme, F., Quezada, Án., & Bustos-Valenzuela, P. (2018). Developing Computational Thinking Skills in Adolescents With Autism Spectrum Disorder Through Digital Game Programming. *IEEE Access*, *6*, 63880–63889. IEEE Access. https://doi.org/10.1109/ACCESS.2018.2877417

Neurodiversity and other conditions. (n.d.). ADHD Aware. Retrieved October 10, 2021, from https://adhdaware.org.uk/what-is-adhd/neurodiversity-and-other-conditions/

Newman, L., Wagner, M., Knokey, A.-M., Marder, C., Nagle, K., Shaver, D., Wei, X., Cameto, R., Contreras, E., Ferguson, K., Greene, S., & Schwarting, M. (2011). *The Post-High School Outcomes of Young Adults With Disabilities up to 8 Years After High School*. 218.

Nightingale, K. P., Anderson, V., Onens, S., Fazil, Q., & Davies, H. (2019). Developing the inclusive curriculum: Is supplementary lecture recording an effective approach in supporting students with Specific Learning Difficulties (SpLDs)? *Computers & Education*, *130*, 13–25. https://doi.org/10.1016/j.compedu.2018.11.006

Nussbaum, N. L. (2012). ADHD and Female Specific Concerns: A Review of the Literature and Clinical Implications. *Journal of Attention Disorders*, *16*(2), 87–100. https://doi.org/10.1177/1087054711416909

Perinelli, M. G., & Cloherty, M. (2023). Identification of autism in cognitively able adults with epilepsy: A narrative review and discussion of available screening and diagnostic tools. *Seizure: European Journal of Epilepsy*, *104*, 6–11. https://doi.org/10.1016/j.seizure.2022.11.004

Pino, M., & Mortari, L. (2014). The Inclusion of Students with Dyslexia in Higher Education: A Systematic Review Using Narrative Synthesis. *Dyslexia*, *20*(4), 346–369. https://doi.org/10.1002/dys.1484

Rao, S., & Gartin, B. (2003). *Attitudes of University Faculty toward Accommodations to Students with Disabilities*. *25*(2), 8.

Rello, L., & Baeza-Yates, R. (2016). The Effect of Font Type on Screen Readability by People with Dyslexia. *ACM Transactions on Accessible Computing*, *8*(4), 1–33. https://doi.org/10.1145/2897736

Ritvo, R. A., Ritvo, E. R., Guthrie, D., Ritvo, M. J., Hufnagel, D. H., McMahon, W., Tonge, B., Mataix-Cols, D., Jassi, A., Attwood, T., & Eloff, J. (2011). The Ritvo Autism Asperger Diagnostic Scale-Revised (RAADS-R): A Scale to Assist the Diagnosis of Autism Spectrum Disorder in Adults: An International Validation Study. *Journal of Autism and Developmental Disorders*, *41*(8), 1076–1089. https://doi.org/10.1007/s10803-010-1133-5

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, *13*(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

Rutter, L. A., Howard, J., Lakhan, P., Valdez, D., Bollen, J., & Lorenzo-Luaces, L. (2023). "I Haven't Been Diagnosed, but I Should Be"—Insight Into Self-diagnoses of Common Mental Health Disorders: Cross-sectional Study. *JMIR Formative Research*, *7*, e39206. https://doi.org/10.2196/39206

Sarrett, J. C. (2016). Biocertification and Neurodiversity: The Role and Implications of Self-Diagnosis in Autistic Communities. *Neuroethics*, *9*(1), 23–36. https://doi.org/10.1007/s12152-016-9247-x

Schreffler, J., Vasquez III, E., Chini, J., & James, W. (2019). Universal Design for Learning in postsecondary STEM education for students with disabilities: A systematic literature review. *International Journal of STEM Education*, *6*(1), 8. https://doi.org/10.1186/s40594-019-0161-8

Scott, S. S., Mcguire, J. M., & Shaw, S. F. (2003). Universal Design for Instruction: A New Paradigm for Adult Instruction in Postsecondary Education. *Remedial and Special Education*, *24*(6), 369–379. https://doi.org/10.1177/07419325030240060801

Shaw, R., & Lewis, V. (2005). The impact of computer-mediated and traditional academic task presentation on the performance and behaviour of children with ADHD. *Journal of Research in Special Educational Needs*, *5*(2), 47–54. https://doi.org/10.1111/J.1471-3802.2005.00041.x

Shmulsky, S., Gobbo, K., Donahue, A., & Klucken, F. (2021). Do Neurodivergent College Students Forge a Disability Identity? A Snapshot and Implications. *Journal of Postsecondary Education and Disability*, *34*(1), 53–63.

Smith, C., & Hattingh, M. (2020). *Assistive Technologies for Students with Dyslexia: A Systematic Literature Review* (pp. 504–513). https://doi.org/10.1007/978-3-030-63885-6_55

Smith, F. G. (2012). Analyzing a College Course that Adheres to the Universal Design for Learning (UDL) Framework. *Journal of the Scholarship of Teaching and Learning*, 31–61.

Sorva, J., Karavirta, V., & Malmi, L. (2013). A Review of Generic Program Visualization Systems for Introductory Programming Education. *ACM Transactions on Computing Education*, *13*(4), 15:1-15:64. https://doi.org/10.1145/2490822

Spooren, P., Brockx, B., & Mortelmans, D. (2013). On the Validity of Student Evaluation of Teaching: The State of the Art. *Review of Educational Research*, *83*(4), 598–642. https://doi.org/10.3102/0034654313496870

Stephanidis, C., Akoumianakis, D., & Savidis, A. (2001). *Universal Design in Human-Computer Interaction* (pp. 741–745).

Strauss, A. L., & Corbin, J. M. (1998). *Basics of qualitative research: Techniques and procedures for developing grounded theory* (2nd ed). Sage Publications.

Students With Disabilities. (2023, May). The Condition of Education 2023. https://nces.ed.gov/programs/coe/indicator/cgg

Syntax Coloring. (n.d.). Retrieved July 9, 2024, from

https://eclipse.dev/pdt/help/html/syntax_highlighting.htm

Taylor, M., Duffy, S., & Hughes, G. (2007). The Use of Animation in Higher Education Teaching to Support Students with Dyslexia. *Education & Training*, *49*(1), 25–35. https://doi.org/10.1108/00400910710729857

Taylor, M., Turnbull, Y., Bleasdale, J., Francis, H., & Forsyth, H. (2016). Transforming support for students with disabilities in UK Higher Education. *Support for Learning*, *31*(4), 367–384. https://doi.org/10.1111/1467-9604.12143

Types of Color Vision Deficiency | National Eye Institute. (n.d.). Retrieved June 1, 2024, from https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/color-blindness/types-color-vision-deficiency

Understanding AND Accommodating Students with Depression IN THE Classroom. (n.d.). https://doi.org/10.1177/004005990704000106

Van Hees, V., Moyson, T., & Roeyers, H. (2015). Higher Education Experiences of Students with Autism Spectrum Disorder: Challenges, Benefits and Support Needs. *Journal of Autism and Developmental Disorders*, *45*(6), 1673–1688. https://doi.org/10.1007/s10803-014-2324-2

Végh, L., & Stoffová, V. (2017). Algorithm Animations for Teaching and Learning the Main Ideas of Basic Sortings. *Informatics in Education*, *16*(1), 121–140. https://doi.org/10.15388/infedu.2017.07

Vincent, J., Potts, M., Fletcher, D., Hodges, S., Howells, J., Mitchell, A., Mallon, B., & Ledger, T. (2016). 'I think autism is like running on Windows while everyone else is a Mac': Using a participatory action research approach with students on the autistic spectrum to rearticulate autism and the lived experience of university. *Educational Action Research*, *25*, 1–16. https://doi.org/10.1080/09650792.2016.1153978

Web Content Accessibility Guidelines (WCAG) 2.1. (2023, September 21). https://www.w3.org/TR/WCAG21/

Wells, J., Barry, R. M., & Spence, A. (2012). Using Video Tutorials as a Carrot-and-Stick Approach to Learning. *IEEE Transactions on Education*, *55*(4), 453–458. IEEE Transactions on Education. https://doi.org/10.1109/TE.2012.2187451

White, S. W., Elias, R., Salinas, C. E., Capriola, N., Conner, C. M., Asselin, S. B., Miyazaki, Y., Mazefsky, C. A., Howlin, P., & Getzel, E. E. (2016). Students with Autism Spectrum Disorder in College: Results from a Preliminary Mixed Methods. *Research in Developmental Disabilities*, *56*, 29–40. https://doi.org/10.1016/j.ridd.2016.05.010

Zhang, D., Landmark, L., Reber, A., Hsu, H., Kwok, O., & Benz, M. (2010). University Faculty Knowledge, Beliefs, and Practices in Providing Reasonable Accommodations to Students With Disabilities. *Remedial and Special Education*, *31*(4), 276–286. https://doi.org/10.1177/0741932509338348

Zubair, M. S., Brown, D. J., Hughes-Roberts, T., & Bates, M. (2021). Designing accessible visual programming tools for children with autism spectrum condition. *Universal Access in the Information Society*. https://doi.org/10.1007/s10209-021-00842-y