**Portfolio Project: What I Learned in Programming 1**

Andrew Barnes

Colorado State University Global

CSC320-1

Dr. Schwartz

10/8/2023

**Portfolio Project: What I Learned in Programming 1**

Working through the Programming 1 course at Colorado State University Global has been a blast. I was unsure what to expect, but a new world has opened up. I learned about programming, algorithms, variables, decision control, loops, Arrays and ArrayLists, modularization, exception handling, inputs/outputs, and object-oriented principles. Each module showed me how to understand and implement these criteria. Below is a glimpse of my journey with this class.

**Programming & Algorithms**

Programming 1 began with the fundamentals of computer programming. I learned how to think about human ideas and how they can be manipulated into languages that machines understand. It was very important to understand that programs revolve around algorithms, and the better structured the algorithm, the more efficient the program. These algorithms made me understand how important clarity and precision are when communicating with a computer. After all, a machine is only as smart as its programmer.

**Variables & Data Types**

Programming is not just about the algorithm; it’s how a computer programmer commands the declaration of variables. This process involves considering whether a variable should contain a byte, short, int, long, float, double, boolean, or char. Thinking ahead can be important when you must ensure too much memory isn’t being used. In such a case, using a double over and int might be better. Everything depends on the application and what the application is being executed on. Learning variables and different data types made me much more prepared for more intricate ideas in later modules.

**Decision Control Structures**

When this week happened, this module was my favorite so far. I found use with dynamic programming using decision control structures to avoid linear programming. More things become possible as a programmer who can create a control flow. This phase is the breakaway from Hello World programs. No longer will the programs I make print out the same things repeatedly. Now, they obtain logic and alternate directions. This comes in handy when the program grabs user input. As a programmer, I decide what happens with particular user input. This aspect is notably crucial for my future career.

**Repetition Loops**

I’ve recently created an algorithm to count the number of rectangles within a given grid of rectangles. What was so fun about it was what I learned to do with loops from this module. I utilized nested loops where my outer loop was a for loop that iterated the width, and my inner loop was a for loop that iterated through the height. Learning about while loops was fun, too, because of how the program seemed to last forever. Programmers can add a line to a while loop to break away from the loop, causing the program to end. In the terminal, this could be something like the character ‘q’ for quit, or a real application would end the loop when selecting the ‘x’ at the top right of the application. It was a connection I held near and dear when writing code in a Java program's main function.

**Arrays & ArrayLists**

The nice thing about arrays is that real-world problems can be implemented with them. Arrays and ArrayLists are the perfect match for larger data sets. The difference between the two was an eye opener to how many possibilities of programs there are out there. With Arrays being static, they remain a fixed size. They can store primitives and objects, while ArrayLists are dynamic. ArrayLists store objects while providing other functionality like adding and removing elements at runtime. This reminded me a lot about how Python has the append method to add elements to a list. The addAll function in Java was interesting because I could add all of the elements of one ArrayList to another. That being said, Java is much more versatile because the programmer has more options for control. This has pros and cons because the uneducated programmer could miss a few steps if they are used to higher-level languages. In contrast, the fluent Java programmer can work their programmatic wizardry.

**Modularization & Exception Handling**

Methods showed me the importance of modularizing code. The benefits I’ve reaped have led to more clear, reusable code. When creating methods, exception handling is of utmost importance because programmers often use general methods in many different scripts and programs. These programs have to be ready for anything. What if there are too many parameters? What happens if there is an integer argument that was supposed to be a string? When encountering these types of errors, the programmer can proactively write exceptions that anticipate such errors. Rather than dealing with unknown errors at compilation time, when the program is running, an error could print out a message expressing exactly the failure. Another benefit is that the program won’t stop running. The user can chug along while still doing what they must do with the program, except for what triggered the exception.

**I/O & Memory**

All the modules before this one were enlightening, but then I learned I could interact with files. This was mindblowing because files are a part of the computer’s operating system. How can this even be possible? The answer was much simpler than expected. Before my time, programmers worked out functions to work with files in Java. This means I had to write minimal code to export a file from my experience with a program. This is fitting for now, but I think I’ll explore programs written in collaboration with operating systems in the near future. This was an outstanding module of influence and motivation to tinker with real-world applications later.

**Object-Oriented Principles**

There are a couple of programming languages that have object-oriented capabilities. Java is one of those languages. I learned that classes are the blueprint of object initialization and other things like inheritance and polymorphism. Inheritance is when a class derives from other classes that already exist, and polymorphism is the ability of a function or method to be utilized in different ways with the object it is acting on. The most important thing I learned about polymorphism is that it allows objects of different classes to be treated as common objects of a single superclass. This ensures a more modular and flexible approach to programming with object-oriented intentions.

**Conclusion**

I understand what Java code looks like now, and I have some solid guesses on the backend of the software I use at work. Working with Java programming has improved my ability to look at a program and guess what the code looks like, regardless of the language used. Understanding what exists already is a gift that I am grateful for. This transparency is a tool I’ll forever have to form the basis of my programmatic experiences. I’m excited to find opportunities out there with my newfound Java skills.