Lessons Learned from Programming 1

The Key takeaways from Programming 1 include understanding variables, data types, decision and repetition structures, arrays, methods, memory management, classes, and object-oriented concepts. These topics have helped me grasp essential programming skills such as using pseudo code, working with arrays, implementing object-oriented concepts, and more. Lastly, we also learned about the benefits of agile programming.

In the first week of the programming class, we focused on variables and data types to store and manipulate data. We learned about primitive data types like short, byte, long, float, Boolean, and char, each used based on the type and size of data required in the program. For instance, numerical values were stored in double or float data types, while character values were stored in Strings or the char data type.

Moving on to the second week, we delved into decision and control structures, which demonstrated branching in programming. We used keywords like switch and if-else statements to control the flow of the program based on certain conditions. These concepts are crucial for implementing logic in various scenarios, such as traffic control systems and scheduling algorithms.

Moreover, one of the best lessons learned in this course was the concept of repetition control structure also known as loops. The benefits of loops -whether it’s a for-loop, do-while loop, or a while loop-is to reduce redundancy in programming. One line of code may be equivalent to more than 1000. As a result, the program is easier to debug, and memory management is made easier. Consequently, organizations can save energy and costs due to efficiency gains from memory allocation (Gaynor).

Memory is such a big issue, that the white house has recommended the use of memory-safe programs such as Java, Kotlin, Python, Rust, and many others. When computers and the internet came to prominence in the 2000s, many companies were eager to scale production and earned vast profits unlike any time before. Unfortunately, security took a back seat, and a lot of memory was exploited in unsecured computer programs. Fortunately, today security is becoming a key aspect of computer science due to international and local cybersecurity ordinances (Di Molfetta).

Memory allocation is also important to data structures such as arrays and lists. By using these data structures, our world has been able to enjoy the use of real-time information such as search engines. In addition, data structures implementing arrays and lists, have contributed and aided in modern technology such as cloud storage technology or even streaming services recommendations. Moreover, our shopping habits have gone digital, and humanity can now search for products at the click of a button (Chan 93-98).

In the final weeks of the class, we learned object-oriented programming(OOP) concepts. These concepts include inheritance, abstraction, encapsulation, and polymorphism. To begin with, however, classes were also implemented to use OOP concepts to create objects that were used to manipulate and display the data. Classes are used to contain data and methods to create objects-which are digital versions of real-life objects (i.e. a chair).

Speaking of objects, inheritance is very important when it comes to implementing OOP concepts. Inheritance allows a class to inherit another class’s features. These features may include methods and fields. As a result, inheritance enables the use of the same code and the avoidance of redundancy. As discussed earlier, the less code used the more efficient and safer the program is (GeeksforGeeks).

Polymorphism is another concept in computer science that aids in code reusability and memory efficiency. The print command in Java can be overridden by the String method to format a code display in a more readable and useful manner. For example, inside the airport, tracking one’s flight path is crucial and thus needs to be in legible human language instead of machine code. Therefore, polymorphism aids in method overriding and overloading simultaneously leveraging the limited amount of memory available (GeeksforGeeks).

Abstraction is another concept that only allows the user in a program to focus on the important details. For example, if anyone is reading this essay, only the message conveyed is displayed-the relevant information. The technology behind the scenes -irrelevant information such as how the pixels are illuminating the screen, the digital circuits transitioning from on to off states nor the machine code translated in the Java virtual machine are displayed. Instead, such information is hidden and only the relevant information is not abstracted (GeeksforGeeks).

Speaking of information hiding, encapsulation embodies the phrase perfectly. Although like abstraction, encapsulation typically involves protecting data from being accessed unethically. As mentioned earlier, cybersecurity is a big issue in the digital age. One of the ways to mitigate such issues is to sanitize data via the use of encapsulation. For example, back accounts are private only to the authorized and verified user. Behind the scenes, each account is its private class available only to the legitimate user. Therefore, encapsulation shields and protects user data, whereas abstraction provides only the necessary details that a user needs (GeeksforGeeks).

Finally, we learned that Agile programming offers benefits such as flexibility to adapt to change, faster delivery of working software, improved collaboration and communication, and enhanced customer satisfaction by delivering value in each iteration, thereby enhancing overall project success (Kavunenko).

In conclusion, the lessons learned in this course showed how to distinguish data types. In addition, the use of data structures and algorithms enabled us to apply programming constructs for reading and writing text files, creating interfaces to perform certain actions, and demonstrating the power of OOP concepts to create portfolio projects. Thus, the lessons learned in Programming 1 have equipped me with a solid foundation in programming concepts and principles. By applying these skills in practical exercises and projects, I have gained a deeper understanding of programming and the importance of continuous learning and exploration in the field.

**References:**

Chan, Jamie. “Chapter 7: Object Oriented Programming Part 1.” *Learn Java in One Day and Learn It Well*, 2016, pp. 93–98.

“Difference between Abstraction and Encapsulation in Java with Examples.” *GeeksforGeeks*, GeeksforGeeks, 12 Mar. 2024, www.geeksforgeeks.org/difference-between-abstraction-and-encapsulation-in-java-with-examples/?ref=lbp.

DiMolfetta, David. “White House Urges Software Developers to Use Memory-Safe Programming Languages.” *Nextgov.Com*, Nextgov/FCW, 26 Feb. 2024, www.nextgov.com/cybersecurity/2024/02/white-house-urges-software-developers-use-memory-safe-programming-languages/394455/.

Gaynor, Alex. *What Is Memory Safety and Why Does It Matter?*, PROSSIMO, 14 June 2021, www.memorysafety.org/docs/memory-safety/.

Kavunenko, Olga. *Guide to System Development Life Cycle*, 2023, https://www.intellectsoft.net/blog/what-is-system-development-life-cycle/#:~:text=SDLC%20comprises%20seven%20different%20stages,testing%2C%20implementation%2C%20and%20maintenance. Accessed 18 Apr. 2024.