Milestone Two – Software Design and Engineering

Christian Bass

SNHU

Artifact:

This artifact is a collection of projects from “IT 145: Foundation in Application Development”. It includes four files, Zoomonitor.java, habitats.txt, animals.txt, and TextMsgExpander.java. All this code was written a few years ago when I initially took the course. Zoomonitor.java is a piece of software I wrote to access the two text files, animals.txt and habitats.txt, and read out the information in them to the user through a menu system. The menu allows the user to select whether they want the information from habitats or animals, and then allows the user to enter the habitat/animal name they would like to view. This code also has the ability to display pop up warning boxes if there is an issue with the animal or habitat denoted by a series of ‘\*’ characters within the text file. TextMsgExpander.java allows the user to input a string and outputs a translated string to the user. This code is capable of expanding common text message abbreviations to their full-length counterparts. An example of this would be changing the string “Hey I’ll BBL” to “Hey I’ll be back later”.

I selected this artifact because it would be a great basis to show how I have improved my skills while earning my degree. This artifact is simple but uses features such as file access and nested loops that would be a good showcase of my skills when refactored to C++.

I improved this artifact by refactoring the entire project to C++ for additional efficiency, optimized how the code was written to be cleaner, more readable, and conform to coding standards, and added further object-oriented design for easier maintainability.

Course Outcomes:

My improved artifact meets four of the five possible course outcomes:

**Outcome one: Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision making in the field of computer science**

For this outcome I created clean, well commented code that maintains readability and conforms to C++ coding standards. This commenting is visible throughout all of my code and denotes all functionality and variables. I strove to format my code to be as readable and maintainable as possible by using object-oriented programming and proper indentation standards.

**Outcome two: Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution, while managing the trade-offs involved in design choices**

I reached this outcome in Two ways. The first was by using pseudocode to outline my planned enhancements before creating the code. Using pseudocode allowed me to solve logic issues in the planning phase leading to a cleaner, more functional implementation and satisfying the outcome requirements further.

**Outcome four: Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals**

Throughout my work on this project I used iterative testing techniques by implementing test code for each function and ensuring its functionality before continuing development. This allowed me to create functional bug free code in less time than would be possible with other testing strategies. I used industry standard software design strategies throughout my code. I used proper commenting and formatting to denote functionality, made it readable and maintainable by using object oriented programming, used proper naming conventions, and formatted the code to industry standards.

**Outcome five: Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources**

For outcome five I worked to ensure that my code was bug free and had no major security vulnerabilities. I avoided the use of any deprecated functions and limited inclusion of third-party libraries to what was strictly necessary. I also made sure that my code was maintainable and expandable by using object-oriented programming and best practices to ensure readability.

I did meet all the original outcomes I had planned to and did not have to make any major changes to do so.

Reflection:

I ran into many issues throughout the process of refactoring my code. The first major issue I had was in the Zoomonitor portion of the project. As I was working on the file access portions of the code, I had to change the design used due to differences between javas scanner, and C++s fstream. The next issue I ran into was with my original code. I had not done a good job of making the code readable and this led to difficulty in interpreting it for the refactoring. Lastly, I had to find a new way to handle strings. In java it was very simple to do tasks such as changing a string to be lowercase, but C++ uses different logic that was more complex to implement.

Pseudocode:

**A screenshot of a computer program

Description automatically generated**

A screenshot of a computer program

Description automatically generated