3-2 Milestone 2: Enhancement One Narrative

Richard VanSkike

Southern New Hampshire University

CS 499: Computer Science Capstone

Professor Brooke Goggin

Artifact Overview

Enhancement one focuses on the principles of software design and engineering, where I worked to develop and showcase my ability to design scalable, maintainable software solutions aligned with industry standards. The artifact selected for this enhancement is the Grazioso Salvare Animal Rescue database, originally created during my IT-145 course, Foundation in Application Development. This database was designed to track rescue animals and manage their information. The original project, written in Java, marked an early milestone in my programming journey, introducing me to key concepts such as object-oriented programming (OOP), data structures, and basic algorithmic logic.

While the project initially focused on implementing a functional solution, the real learning came from understanding how foundational programming concepts like inheritance, encapsulation, and abstraction, were applied to solve real-world problems. These principles were not just technical requirements but critical to building a maintainable system that others could understand and modify. Over time, this artifact has become more than just an assignment; it has served as a model to show the growth of my programming skills and how I developed a collaborative mindset in designing software.

Justification for Inclusion

I chose this artifact for my ePortfolio because it illustrates the progress of my understanding of software development principles, such as the transition from learning to doing. By translating the project from Java to Python, I demonstrated proficiency in two programming languages and showcased my ability to adapt and improve upon an existing solution. This transition also improved the maintainability and readability of the project, making it more accessible for future users and maintainers, which directly aligns with creating environments for diverse audiences.

For instance, moving methods from the original Driver.java file into separate classes during the refactor solidified my understanding of modularity and helped make the code easier for future developers to navigate. This decision wasn't just about improving architecture; it was also about making the code more collaborative, enabling future developers to easily extend or modify it without needing to sift through large, monolithic files. This restructuring emphasized the importance of clean architecture and its impact on scalability and team collaboration.

Implementing inheritance in Python allowed me to reduce code redundancy by creating more flexible, reusable structures. For example, refactoring the Intake method to eliminate duplicated code not only optimized the program but also made it easier for others to follow and maintain. This effort to make the code more readable and adaptable for different teams or individuals contributes to the course outcome of building collaborative environments.

Improving documentation was another key enhancement. Initially, the Java version of the project lacked detailed comments, which would have made it difficult for others to understand my work. Through the process of translating and enhancing the project in Python, I focused on writing clear and comprehensive comments that would help future developers or non-technical team members understand the purpose and functionality of each part of the code. This experience underscored the critical role that clear communication plays in ensuring smooth collaboration, especially when different team members need to work on the same project.

Achievement of Course Outcomes

For this course I am striving to reach five course outcomes. They are as follows:

- Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision making in the field of computer science.
- Design, develop, and deliver professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts.
- 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.
- 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.
- 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.

With this enhancement I achieved two of these critical course outcomes that demonstrate my growth in both technical and professional skills.

To employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science, I improved the readability and flexibility of the Python version. This ensured that future developers would have an easier time collaborating on the project. Python's syntax is known for its readability, which was a deliberate choice to make the code more accessible and foster collaboration. This transition was not just about technical improvement but also about making the project more collaborative by creating an environment where the code is easy for anyone to understand and modify. Python’s syntax is designed to be clear and straightforward, reducing the learning curve for new developers. This choice promotes inclusivity by making the codebase accessible to a wider audience, including those who may not have extensive programming experience. Python’s dynamic nature allows for rapid prototyping and iterative development, which can be crucial in collaborative settings where requirements may evolve.

I focused on designing, developing, and delivering professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts. The key to this outcome was the refinement of my communication skills through better documentation. The original Java project lacked detailed comments, which taught me firsthand how difficult poor documentation can make future collaboration. By focusing on better commenting practices in Python, I made the project easier to maintain and ensured that future users, whether developers or non-technical stakeholders, could easily understand the code. Good documentation practices, including detailed comments, help bridge the gap between technical and non-technical stakeholders. This ensures that everyone involved in the project can understand the code’s purpose and functionality. Beyond comments, creating comprehensive documentation, such as README files and user manuals, will further enhance understanding and collaboration.

Reflection on the Enhancement Process

This project taught me that programming is much more than just writing code; it’s about problem-solving, adapting to new challenges, and continually learning. Translating the program from Java to Python provided me with a deeper understanding of both languages, but more importantly, it revealed gaps in my knowledge that I was able to address during the enhancement process.

One of the major challenges was understanding how to properly implement inheritance in Python. Java has strict rules for constructors and inheritance, whereas Python offers more flexibility. This required a shift in mindset, and through research on platforms such as GeeksforGeeks (2020) and Real Python (2023), I was able to deepen my understanding of object-oriented programming. By leveraging these resources, I also gained experience in how to seek help and learn from others, which is a crucial part of collaboration in software development.

Another challenge I faced was working with Python's data structures, specifically arrays. Through research on sites such as w3schools.com (n.d.) and revisiting these core programming concepts reinforced the importance of staying familiar with the fundamental concepts, even after a long time away from them. This challenge taught me that ongoing learning is vital in the field of computer science and that the ability to quickly adapt to new languages is crucial for collaboration and long-term success.

In the end, enhancing this artifact strengthened my technical foundation and reminded me of the importance of designing software not just to work, but to be understandable, scalable, and ready for collaboration. These enhancements not only improved the artifact itself but also helped me meet the course outcomes related to building collaborative environments and delivering professional-quality communications.

REFERENCES

GeeksforGeeks. (2020, August 1). *Calling a super class constructor in Python*. https://www.geeksforgeeks.org/calling-a-super-class-constructor-in-python/

Real Python. (2023, December 22). *Getters and setters: Manage attributes in Python*. https://realpython.com/python-getter-setter/

W3schools.com. W3Schools Online Web Tutorials. (n.d.). https://www.w3schools.com/python/python_arrays.asp