Timothy McIntire

CS 499

Professor Bryant (Professor Hawk)

May 25, 2025 (revised June 18, 2025)

Milestone Two Narrative

1. Briefly describe the artifact. What is it? When was it created?

The artifact I enhanced is my final project from IT 145: Foundations in Application Development, developed in early 2023. The original project is a Java-based command-line application simulating an animal intake and reservation system for a fictional rescue organization, Grazioso Salvare. Initially, the application only supported two animal types—dogs and monkeys—each hardcoded as their own classes. A single Driver class handled all logic, including intake, search, reservation, and listing of animals.

2. Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components showcase your skills and abilities in software development? How was the artifact improved?

I selected this artifact because it represents a foundational project in my Computer Science journey, and it provides a perfect platform to showcase my growth in software engineering and modular design. The original code was functional but limited in flexibility and maintainability. By enhancing it, I was able to demonstrate advanced object-oriented programming (OOP) skills, including:

- Abstraction by refactoring the RescueAnimal class into an abstract superclass
- Inheritance by creating six new animal “kingdom” classes (Mammal, Bird, Reptile, Amphibian, Fish, Other)
- Encapsulation through proper use of private fields and public getters/setters
- Modular design by eliminating duplicate menu logic and using polymorphism for handling all animal types

I also improved scalability by replacing the species-specific class structure (e.g., Dog, Monkey) with broader kingdom classes that allow flexible animal entry without changing the program structure. A new family field allows separation of animals like dogs, cats, monkeys, etc., within each kingdom. Furthermore, I implemented a new ID system using prefixes (like MAM001, BIR001) that ensures each animal has a unique and scalable identifier.

Additionally, as part of improving security and reliability, I began analyzing potential vulnerabilities in my program design. For example, I recognized that relying solely on text-based user input without strict validation could allow malformed entries—such as negative ages, blank fields, or duplicate IDs. While I implemented basic validation (ensuring unique IDs and parsing numeric values), I also plan to expand this by introducing regular expression checks, range validation for physical attributes (like weight), and more robust error handling. These steps lay a foundation for future enhancements, including sanitization techniques to mitigate injection attacks in later database-connected versions.

3. Did you meet the course outcomes you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?

Yes, I fully met the course outcomes I planned for Category One. These include:

- CO1 (Collaborative Software Design): I applied modular OOP principles to create a scalable and maintainable system. The code is now easy to understand, extend, and collaborate on.
- CO3 (Computing Solutions and Design Choices): I redesigned the class hierarchy to simplify future enhancements and clearly manage trade-offs between flexibility and complexity.
- CO5 (Secure and Reliable Systems): I began implementing input validation for key fields and structured the program to prevent duplicate or invalid animal entries.

There are no updates needed to my original plan. The enhancements have laid a strong foundation for the next two categories: data structure optimization and database integration.

4. Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?

Enhancing this artifact taught me a great deal about designing for scalability and maintainability. I realized how limiting it was to create new hardcoded classes for each animal type, and how much more powerful it is to use abstract classes and polymorphism.

The biggest challenges were:

- Managing complexity while refactoring: As I removed the species-specific classes and restructured the hierarchy, I had to ensure nothing broke in the intake or reservation process.
- Ensuring consistent user experience: I redesigned the menu system to be clean, logical, and easily extendable as new animal types are added.

- ID generation and input validation: I added scalable IDs and began enforcing valid input formats (ensuring reserved status is boolean and age is a number). However, I also reflected on how vulnerabilities could arise if stricter validation isn't enforced, like accepting empty names or improperly formatted fields. To address this moving forward, I plan to incorporate regex-based field validation, enforce character constraints, and eventually centralize input sanitization logic to improve reliability and reduce the risk of unexpected behavior.

These changes significantly improved the quality of the system and demonstrated my growing understanding of clean architecture, OOP, and real-world software design principles.

Conclusion

This milestone demonstrates my ability to refactor and improve a previously basic system using modern software engineering practices. The new version is cleaner, more modular, and prepared for future enhancements in performance and data persistence. I’m confident that this work aligns with industry expectations for scalable, professional software systems and directly supports my career goal of becoming a software developer or engineer.