**3-2 Milestone: Enhancement One: Software Design and Engineering**

Lauren-Ann Javier

Southern New Hampshire University

CS 499

Mr. Martinez

07/17/2025

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

The artifact is a Java-based Contact Management system consisting of 4 classes, Contact, ContactService, ContactTest, and ContactServiceTest. This artifact was originally created during my CS 320 class and has been progressively enhanced for this CS 499 Capstone project. Contact represents an individual contact with validated fields for contact ID, first name, last name, phone, and address. ContactService manages these contacts by supporting add, update, delete, and retrieval operations. The test classes provide comprehensive unit testing to make sure it is strong and error-free functionality.

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

I selected this artifact because it demonstrates my ability to design and implement object-oriented systems that enforce data integrity through validation, encapsulation, and exception handling. The Contact class showcases careful field validation to prevent invalid data entry, an essential software engineering practice. The ContactService class exemplifies managing collections of objects, throwing meaningful exceptions when operations violate business rules, for example, adding duplicate contacts or updating non-existent contacts. Together, these classes show my skills in software design, error handling, and maintaining data consistency. The accompanying test classes demonstrate my competence in writing unit tests with JUnit 5, covering positive and negative test cases thoroughly, which is critical for professional-quality software development.

1. **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?**

Enhancements I made for this milestone:

* **Contact class:** Added detailed JavaDoc comments for all methods and fields to improve code readability and maintainability. Improved exception messages to be more descriptive and consistent.
* **ContactService class:** Added input validation and explicit exception throwing for invalid operations, such as deleting or updating a non-existent contact, aligning behavior with my test expectations. Included validation for contactId parameters to prevent null or empty inputs. Improved JavaDoc documentation by adding comments.
* **ContactTest and ContactServiceTest:** Added descriptive assertion messages in unit tests to clarify the intent and aid debugging. Expanded tests to cover additional edge cases such as null inputs and invalid field lengths. Improved test method naming for clarity. Added JavaDoc-style comments to each test method explaining its purpose.

These enhancements significantly improved the strength, clarity, and professional quality of the code, making it more maintainable and easier to understand by other developers.

Some course outcomes that were addressed in this artifact included:

* Designing and evaluating computing solutions that solve a problem using software engineering principles (validation, exception handling).
* Demonstrating the ability to implement well-structured, professional-quality code with clear documentation and testing.
* Employing strategies for ensuring software quality through comprehensive unit testing.
* Developing a security mindset by validating inputs to prevent invalid or bad data from entering the system.

1. **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?**

During the enhancement process, I learned the importance of difficult input validation and how it helps maintain a consistent system state. Writing detailed JavaDoc comments reinforced the value of clear documentation for professional codebases. One challenge was making sure that the service layer’s behavior (throwing exceptions on invalid operations) exactly matched the expectations of my unit tests, which required revisiting method logic and adding input checks. Another challenge was designing comprehensive unit tests that not only check valid operations but also handle all expected invalid inputs and edge cases, improving overall test coverage and software reliability. Overall, this process deepened my understanding of software design best practices, error handling, and the role of unit testing in delivering high-quality software.