**Southern New Hampshire University**

Joshua Wozny

joshua.wozny@snhu.edu

May 28, 2023

**CS-320: Journal: Testing Approach and JUnit**

Software testing is an essential process in the development of software, as it ensures that the final product meets the requirements and is free from errors. J Unit is an unit testing framework for Java that can be used to integrate testing into the development process – especially when combined with an IDE and a build system like Maven.

Unit Tests are designed to ensure software requirements are met. Each unit test tests a specific requirement of a class, method, or function. The goal is to ensure that requirements are met, and bugs are caught during the earliest stages of development. Tests are re-run after all changes to the code to ensure that no breaking changes have been made.

In the most recent exercises, developing the contact and task services, J Unit was used to ensure all tasks, contacts, and services met requirements. Using J Unit was integral to verifying that these work as intended. Each property of the Entities, contact and task, was tested to ensure that valid strings were allowed, and that invalid strings were disallowed. This was accomplished through the use of comparisons (id==null, task.description.length()>50) to throw exceptions for values outside of the expected range. With close to 100% code coverage I am confident that these services function as intended.

I used best practices to ensure technically sound code, using inheritance for Tasks and TaskService (and refactoring Contact and ContactService similarly) by utilizing Entity and EntityService abstract classes, respectively. This helped standardized and simplified how Ids are treated between all Entities as the application grows larger. Keeping each class, including abstract base classes and new Exception classes in separate files within a Maven project keeps the source modular, and organized – improving the re-usability and maintainability of the software. See line 5 of the TaskService to see it’s inheritance as well as the EntityService class for details.

Using the correct data structures helped to make sure that the code is efficient, using HashMap to store contacts and tasks in their associated services. Using the HashMap structure made the Services easier to implement and makes keeping unique values for Ids straightforward since the Id was used as the key of the HashMap, see line 7 in the Task Service, as well as line 21 to see how the HashMap simplified the service implementation.

Using clean coding practices, good commenting, and self-documenting code through naming conventions, the current code base is in good shape and ready to be added to and get integrated into a larger application. As more code is developed, additional refactoring may be needed to ensure that the application can continue to be extended, and will be easy to understood by other developers. The unit tests will also aid other developers to understand the software requirements as they clearly outline what these classes are designed to do.