**7-2 (Project2: Unit Testing)**

**Student Name: Kursheeka Milburn**

**Submitted To: Karl Lewis**

**Introduction**

Software testing is a critical process and ensures the reliability, security, and correctness of an application during software development, in this report we will analyze the functionality of the contact management system through unit testing specifically called Junit. The system we have developed includes different modules for managing contacts, appointments, and tasks, the primary purpose of these tests is to verify the integrity of system logic and prevent invalid data. In this report, a structured evaluation of unit test cases is provided that are developed for different components of the system ensuring compliance with functional requirements.

**Software Testing and Junit Framework**

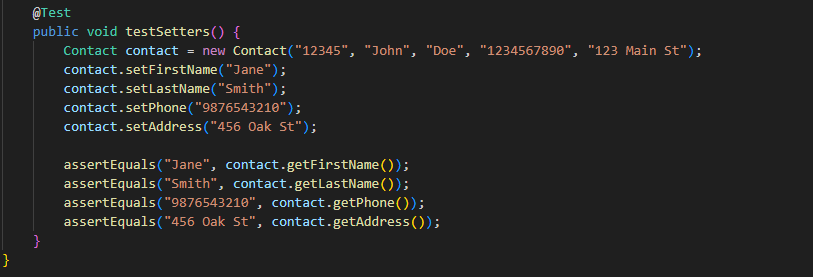
Software testing is a process that involves executing a program to identify bugs and ensure that the developed software fulfills all expected requirements, in unit testing we test each individual component of the code in isolation. We used the Junit test for testing our developed application because Junit is a widely used framework for Java applications that enables developers to create automated test cases and ensure robustness while providing assertions to validate expected outcomes. These tests help to identify and fix issues in the early development phase and facilitate code refactoring which ensures code modification does not produce new errors and speeds up testing and execution.

**Test Cases**

In this section we will discuss the details of the unit testing implemented for contact service, task service and appointment service, this section covers test cases results and expected outcomes which demonstrate object creation validation and exception handling.

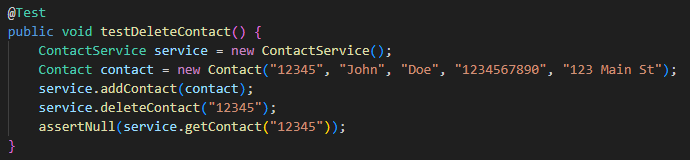
**Contact Test.java**

In this case, we first implemented valid contact creation that ensures all the instances of “Contact” are created correctly with valid inputs. Invalid contact creation verifies that invalid inputs throw exceptions and setter functionality updates on all the fields like phone number, address first name, and last name.



**Contact Service Test.java**

The key test cases include adding contacts, duplicate contact prevention, deleting contacts, and updating contacts. Adding contact ensures that the added contact is successfully retrieved and duplicate contact prevention confirms updates to existing field while deleting contacts check that deleted and non-existent contacts are not accessible.

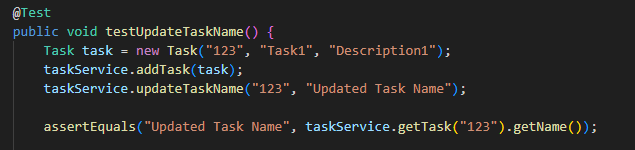


**Task Test.java**

The test cases include valid task creation which ensures that a valid task is created with appropriate constraints, updating task fields that test setters for task name and description and invalid task attributes verify that any null or overlong name and description throw exceptions.

**Task Service Test.java**

The test case verifies that all the added tasks are ready for any additions and retrievals, duplicate task ID handling ensures that adding a task with existing pre-defined ID results in an exception and deleting and updating tasks verifies behavior for field updates and deletion.

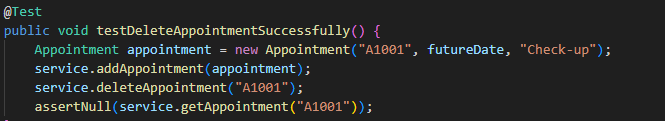


**Appointment Test.java**

The test cases ensure a valid appointment creation that verifies that appointments with valid attributes are successfully created and ensure that null, past dates, over long, and missing descriptions throw exceptions and restrict all invalid descriptions and null values.

**Appointment Service Test.java**

The test appoints management system and adds appointments with unique IDs and prevent duplicate IDs and also delete existing and non-existing appointments and manage error handling for invalid dates.



**Junit Tests Quality**

I believe that all the Junit tests demonstrate comprehensive coverage, clear test organization, and well-structured as all service classes have dedicated test classes and each method has multiple test cases, edge cases, and error conditions that are thoroughly tested.

**Employed Testing Techniques**

The employed testing techniques include unit testing which helped in testing each service method independently and use cases to verify individual components. As mentioned previously in module 6 the “TaskServiceTest” class verifies functionality like updating tasks, deleting, and adding to confirm that all the exceptions are correctly thrown for the duplicates IDS and non-existent tasks it also provides immediate feedback on specified blocks and also allows testing of all types of scenarios. Boundary testing techniques help to ensure that the input parameters meet the specified constraints which helps to identify all off-by-one errors and ensure that the system efficiently handles all maximum and minimum values that are expected to process. Additionally, integration and system performance testing techniques are also deployed at certain levels and ensure the whole system works accurately It also helps to identify those issues that unit tests miss and make it more robust in error handling.

**Conclusion**

The unit test that is conducted on different modules of the contact management system helps us to validate the robustness of the application, these tests highlight constraints violations and ensure system reliability. All the modules implemented are tested for proper input validation, logic consistency, and duplicate prevention, with the help of rigorous testing we ensure that our designed system is well-structured and meets all the requirements and also ensure that the system does not introduce defects in future modifications and scalability.