Matthew Cochrane

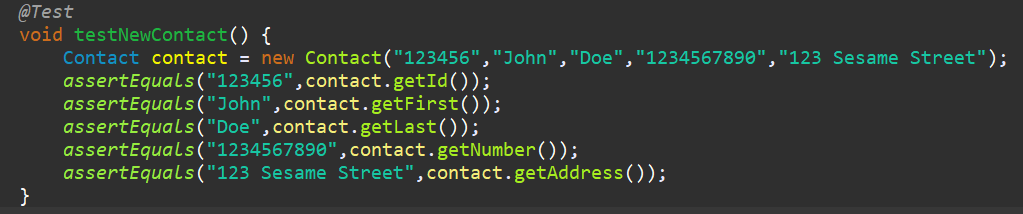
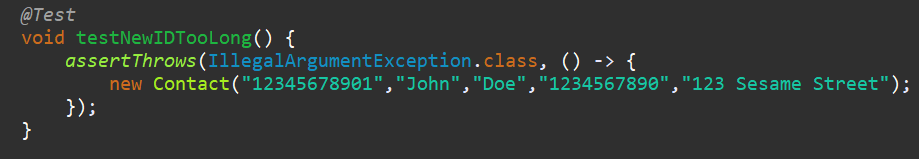
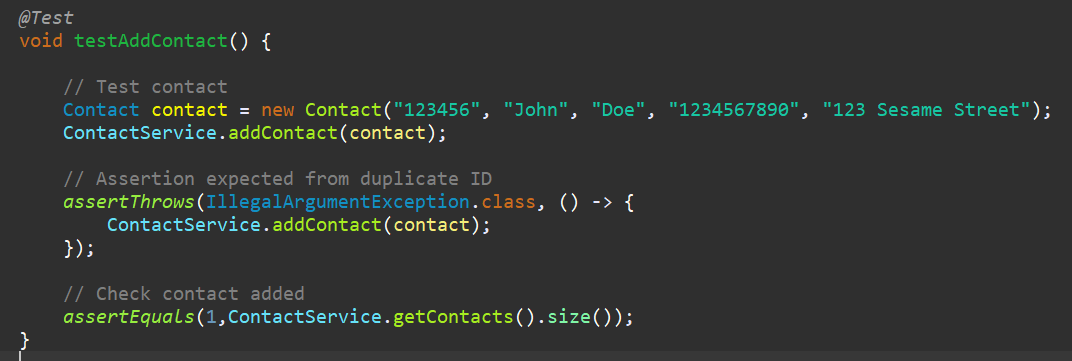
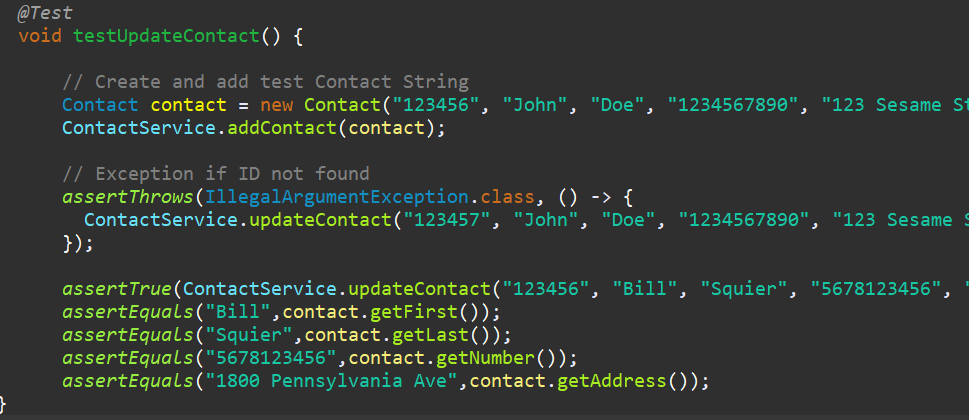
CS 320

12 June 2023

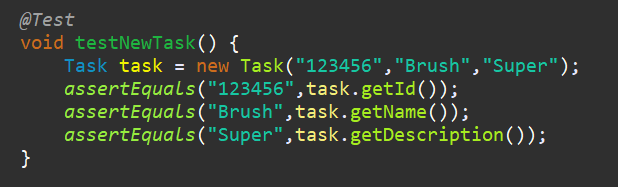
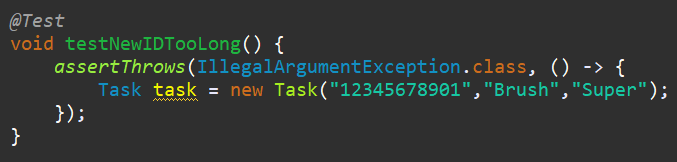
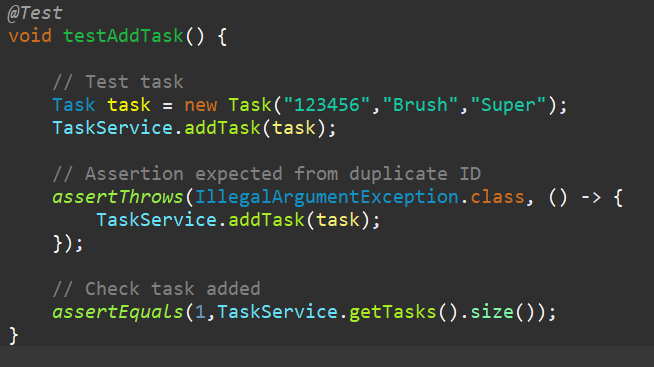
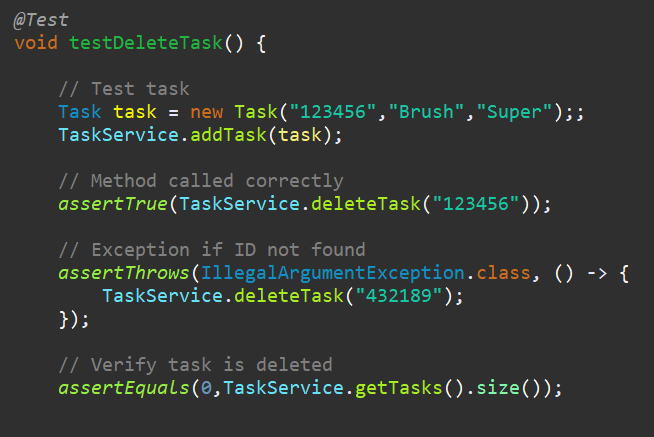
Summary And Reflections Report

1. Summary

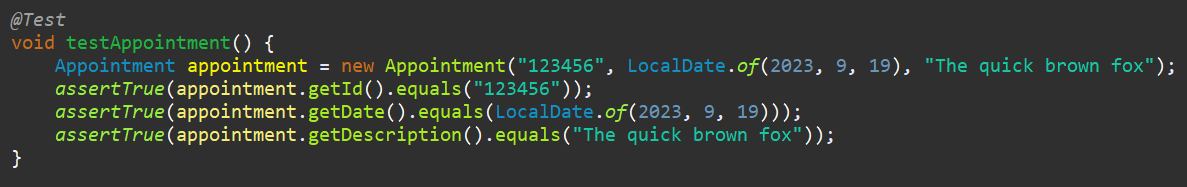
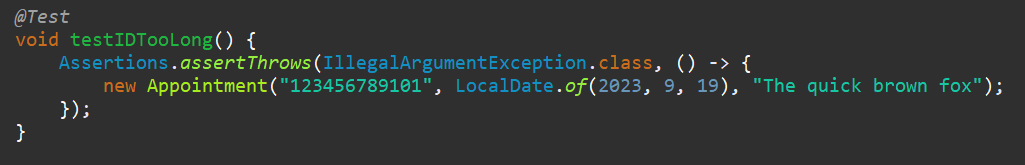
Contact Service

* + ContactTest.java
    - The testNewContact test ensures that a valid contact can be created and verified.  
      
    - Each of the other tests are to ensure that any invalid contact information is correctly flagged. Here is one example of a contact ID being longer than 10 characters:  
      
    - There is a test for each parameter being invalid, similar to the test shown above.
    - All of the tests together ensure that valid contacts are created and invalid contacts are rejected, aligning with the software requirements.
    - Coverage is 65% which is decent but not what I expected or hoped for. The way my tests are set up, I cannot seem to reach some lines in my setter methods. The exceptions are thrown when appropriate though, and that is the most important part of the test.
  + ContactServiceTest.java
    - The testAddContact method throws an exception when a duplicate ID is used, as well as confirms the valid contact is added to the list.  
      
    - The testDeleteContact method throws an exception when a non-existent ID is used, as well as confirms the valid contact is deleted from the list.  
      
    - The testUpdateContact method throws an exception when a non-existent ID is used, as well as confirms each parameter is updated.  
      
    - These tests together ensure that the requirements for contact service are met.
    - Coverage is 83%. The only lines missed are the lines inside the assertThrows bracket, which is expected. Otherwise, there is complete coverage.

Task Service

* TaskTest.java
  + The testNewTask test ensures that a valid task can be created and verified.  
    
  + Each of the other tests are to ensure that any invalid task information is correctly flagged. Here is one example of a task ID being longer than 10 characters:  
    
  + There is a test for each parameter being invalid, similar to the test shown above.
  + All of the tests together ensure that valid tasks are created and invalid contacts are rejected, aligning with the software requirements.
  + Coverage is 74.2%. The only lines that weren’t executed were the lines that would throw an exception, which is expected.
* TaskServiceTest.java
  + The testAddTask method throws an exception when a duplicate ID is used, as well as confirms the valid task is added to the list.  
    
  + The testDeleteTask method throws an exception when a non-existent ID is used, as well as confirms the valid task is deleted from the list.  
    
  + The testUpdateName method throws an exception with an invalid name or ID, as well as verifies the name is updated.  
    
  + The testUpdateDescription method throws an exception with a invalid description or ID, as well as verifies the description is updated.  
    
  + These tests together ensure that the requirements for the task service are met.
  + Coverage is 76.3%. The only lines that weren’t executed were the lines that would throw an exception, which is expected.

Appointment Service

* AppointmentTest.java
  + The testAppointment test ensures that a valid appointment can be created and verified.  
    
  + Each of the other tests are to ensure that any invalid appointment information is correctly flagged. Here is one example of an appointment ID being longer than 10 characters:  
    
  + There is a test for each parameter being invalid, similar to the test shown above.
  + All of the tests together ensure that valid appointments are created and invalid contacts are rejected, aligning with the software requirements.
  + Coverage is 75.8%, I know my tests are effective because exceptions are thrown when they are supposed to be.
* AppointmentServiceTest.java
  + The testAddAppointment method throws an exception when a duplicate ID is used, as well as confirms the valid appointment is added to the list.  
    
  + The testDeleteAppointment method throws an exception when a non-existent ID is used, as well as confirms the valid appointment is deleted from the list.  
    
  + These tests together ensure that the requirements for the appointment service are met.
  + Coverage is 87.1%, which is indicative of quality tests.

I ensured that my code is technically sound by using the Assertions library to verify any changes that the test would make. For example, in this snippet I am updating the description and using the assertEquals() function to verify that the change is made. There is a similar assertEquals() function for each test that needs to check the status of the object after a change is made.

*assertEquals*("Ultra",task.getDescrption());

I know that my code is efficient because each test is doing only one simple task. If the test were more complicated, extra variables are introduced which makes the test extremely inefficient.

*@Test*

void testNewIDTooLong() {

*assertThrows*(IllegalArgumentException.class, () -> {

Task task = new Task("12345678901","Brush","Super");

});

}

1. Reflection
   1. Testing Techniques  
      The type of testing employed in these services is called white box testing. This is a technique where the tester has complete knowledge of the code base.
      1. Decision testing – This is where each branch of a decision statement (IF or CASE) is tested individually.
      2. Statement testing – This kind of test will attempt to execute lines of code and determine various things, such as in the assignment where assert functions are used to verify changes in the hashmap.
      3. Practical uses
         1. Unit testing – Using unit tests will dramatically reduce the amount of errors in the source code.
         2. Coverage analysis – This enables a tester to more easily detects gaps or faults in the test and/or the source code.

Two types of testing are not used, black box and experience.

* + Black box testing is a technique where the tester has zero knowledge of the code base and must rely on logging information and other outside resources.
  + Experience testing is one used by testers with experience in the source code.
  1. Mindset
     1. When writing the tests, I employed caution by ensuring that each function had at least one corresponding test. It is important to have sufficient testing parameters or else there really is not a point to test anything, not just an application. A chain is only as strong as its weakest link.  
        void testInvalidNewDescription() {*assertThrows*(IllegalArgumentException.class, () -> {  
        Task task = new Task("123456","Brush","Super man batman justice league overton window destruction");
     2. To be honest, I was not cognizant of bias when writing these tests. I simply ran them and changed what I needed to in order to get complete coverage.
     3. It is only after completing this assignment that I fully understand the importance of being disciplined in software testing. Every edge case conceivable must be accounted for, or else the chain will break somewhere down the road when a user reaches the edge case. In the future, I will be using JUnit and other recommended testing techniques.