Brandon Kelfstrom

6/16/22

CS 320 Project 2

The testing for project one was designed to test every requirement of the project. Anything listed as a requirement for the code was tested with JUnit. For instance, the requirements stated that the phone number had to be 10 digits long and not null in Contact.java. So the ContactTest.java file includes a general test of the phoneNumber assignment function, a test for length handling, and a test for null handling. JUnit testing took some time to implement properly. Once it was functioning correctly, I read through the test files to make sure that every variable, function, and feature of those functions was addressed in the testing. Each test function was kept as minimal as possible to prevent errors or inconsistent results:

@Test

public void deleteIDNotFound() {

TaskService ts = new TaskService();

// add tasks

Task t1 = new Task("1", "John",

"A description of an account less than 50 chars");

assertEquals(true, ts.addTask(t1));

assertThrows(Exception.class, () -> ts.deleteTask("2"));

}

(TaskServiceTest.java, line 47-57)

This function includes assertEquals and assertThrows. It makes sure the variable is assigned correctly, then makes sure the task is properly deleted. Utilizing assertThrows and throw exceptions makes testing very predictable and if there are any issues with testing there are few areas to check for bugs. This method also made the code efficient. Using the JUnit library for testing keeps the code simple and ensures the speed of the code at run time:

@Test

public void TaskIDNull() {

assertThrows(IllegalArgumentException.class, () -> new Task(null, "john", "Description"));

}

(TaskTest.java, line 39-42)

The main technique I used for testing was assert equals. This ensures that each value is correctly assigned to the variable it is intended for. I also used assertThrows to make sure that exceptions are thrown when invalid entries are assigned to variables. For the automatic ID generation in the appointment service, assertNotNull is used to make sure that null variables are not created. Any test function that required multiple functions to be called tested each function in order to make sure errors did not occur before the actual intended test. This testing technique is useful to pinpoint the area of failure in the program. When writing code, I did my best on my own, but I wanted to keep quality in mind. I looked up ways to implement my ideas better and test code more efficiently. This is a common technique used in the field, and my understanding of testing has significantly improved thanks to all of the resources available online.

For this project, I tried to stick to the rubric for software functionality. The code is intended to address every requirement as efficiently as possible. When testing the software, the rubric was similarly followed, testing every different requirement individually. When creating software, it is important to keep scalability and maintainability in mind. Writing code that is well separated into functions makes future changes easy for not only yourself but any other developers. Caution is very important when creating a complex piece of software. Having multiple files interacting with each other introduces more possible points of failure. Testing to make sure each file functions properly helps make sure that the files will interact as intended. In the appointment service code, I tested the appointment ID generation to make sure that the ID could be called in the appointment.java file without error.

Bias is an important thing to leave behind in all aspects of product design. When creating and testing code, the goal is to make the best code possible. If you are going to be biased when testing, you may as well not test at all. Testing in a way that might make your original code seem better will only come back to bite you if the code was not perfect to begin with. Depending on the development environment, one might feel like they need to forego testing to prevent issues popping up and being reprimanded. Code is objective, and should be treated as such. Some code can be better than others, but if code fails, it is objectively bad.

Discipline is another thing that programmers must maintain when developing code. The quality of code needs to be maintained throughout the program, and testing needs to be done thoroughly to test all aspects of the code. In the field, there will not be a rubric, but there will be a list of software requirements. These requirements need to be followed as rigidly as possible to ensure the quality operation of the end product. Testing should also be done to test every aspect of the program requirements as well.

References

Brandon Kelfstrom. (2022). CS320 Project One (Version 1.0) [Java].