Kendrick Free

Professor Gaston

CS-320

19 August 2022

**Project Two Report**

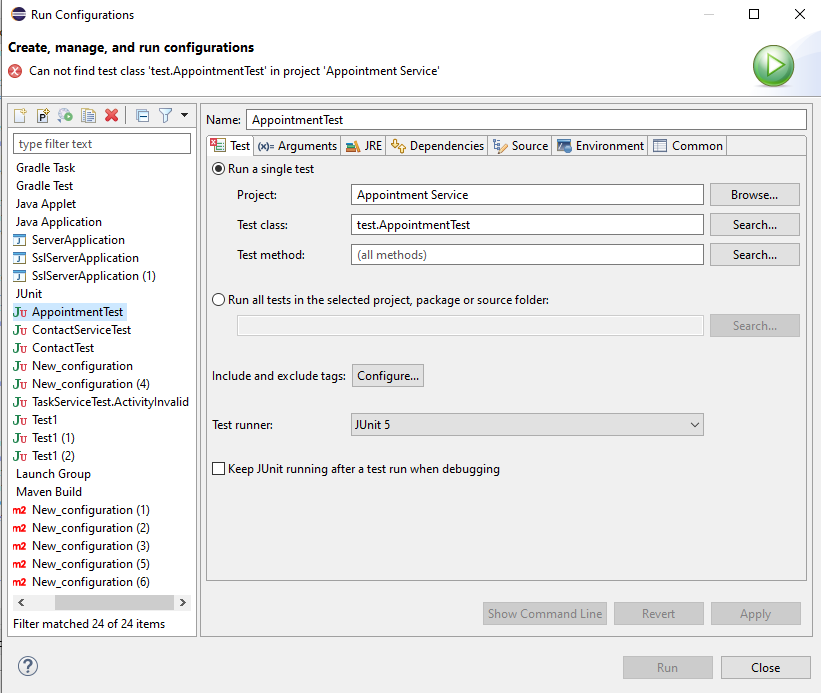
1. **Summary**
   1. **Describe your unit testing approach for each of the three features.**
2. **To what extent was your approach aligned to the software requirements? Support your claims with specific evidence.** 
   * 1. **Defend the overall quality of your JUnit tests. In other words, how do you know your JUnit tests were effective based on the coverage percentage**

This project came together very slowly and tediously. Having three separate parts to handle and get working correctly proved to be quite the task. The most difficult part was coordinating the JUnit test to work correctly and display the correct information. It was successful on two of the three parts of the project and yielded errors that needed to be fixes.

* 1. **Describe your experience writing the JUnit tests.**

1. **How did you ensure that your code was technically sound? Cite specific lines of code from your tests to illustrate.**

Writing the JUnit test was rather difficult, and I had a lot of trouble interpreting the results. Dependency checks are easier to use and perform in my opinion and easier to interpret. I ran Junit test on every part of the project to which only one was successful in showing results and compiling. An included screenshot will show all the JUnit test configurations I attempted to use.



* + 1. **How did you ensure that your code was efficient**? **Cite specific lines of code from your tests to illustrate.**

While my code may not be the most efficient, I am still a novice when it comes to coding and have much left to learn in terms of efficiently. The main class utilizes all the project classes in Contact, Application, and Task. The test in the main class remains the same across all the classes to ensure simplicity and that the same test is run.

Example code:

ContactService contactService = **new** ContactService();

//Contact to use for the test

Contact A1 = **new** Contact("A1", "Kendrick", "Free", "9786451728", "Harker Heights, TX");

Contact A2 = **new** Contact("A2", "Brian", "Johnson", "8765367182", "Cambridge, MA");

//Adding contacts test

**if**(contactService.addContact(A1)) {

System.***out***.println("Contact added.\n");

}

**else** {

System.***out***.println("Contact not added.\n");

}

**if**(contactService.addContact(A2)) {

System.***out***.println("Contact added.\n");

}

**else** {

System.***out***.println("Contact not added.\n");

}

//Add a contact twice test

**if**(contactService.addContact(A2)) {

System.***out***.println("Contact Added.\n");

}

**else** {

System.***out***.println("Contact cannot be added.\n");

}

//print list will display the contact list

contactService.printList();

//Delete contact Test

contactService.deleteContact("A1");

**if**(contactService.deleteContact("A1")) {

System.***out***.println("Contact deleted.\n");

}

**else** {

System.***out***.println("Contact has already been deleted.\n");

}

//Update contact info test

contactService.contactUpdate("A1", "Kendrick", "Free", "9786451728", "Harker Heights, TX");

//print contacts to view any changes

contactService.printList();

1. **Reflection**
   1. **Testing Techniques**
2. **What were the software testing techniques that you employed in this project? Describe their characteristics using specific details.**

The most utilized form of testing used during this project was the JUnit test simply because it was called for in the rubrics. I also read the entirety of the code browsing for any errors the naked eye could catch. I wish I utilized the Dependency check in this project, but I could not figure out how to access the pom.xml file to insert the parameters for the dependency check.

* + 1. **What are the other software testing techniques that you did not use for this project? Describe their characteristics using specific details.**

Aside from the JUnit test and physically viewing all the code I regret not using dependency checks more throughout the project. The dependency check will locate any compilation errors and address them in terms of severity. The only issue I has utilizing the dependency check was that the pom.xml file did not exist for me to insert the check.

* + 1. **For each of the techniques you discussed, explain the practical uses and implications for different software development projects and situations.**

The three techniques I discussed are easily utilized across other types of projects and development cycles. The JUnit test can be utilized in any environment to test for code errors and will return the results will be shows to the user right on the screen. Physically looking through the code is the easiest technique to employ and the most prone to human error. The longer the code the more prone the user is to miss a detail or incorrect character. Lastly, the dependency check can be utilized on almost any kind of project or development cycle if the pom.xml file is present to insert the dependency check parameters.

* 1. **Mindset**

1. **Assess the mindset that you adopted working on this project. In acting as a software tester, to what extent did you employ caution? Why was it important to appreciate the complexity and interrelationships of the code you were testing? Provide specific examples to illustrate your claims.**

While working on the project I was constantly stopping to review the code from the beginning to the point I had stopped at to ensure everything was correct and did what I needed it to do. While this was not the case with all the code, I did employ caution when it came to using certain commands, imports, and fixing errors that I knew how to fix. Some portions of the code had errors that I could not fix no matter what research I did into the issue. Here is one of the issues:

(Calling for addAppointment which is defined in the AppointmentService class, but it still comes up red and causes compilation errors.)

(Main class calling addAppointment)

**if**(AppointmentService.addAppointment("C")) {

System.***out***.println("Appointment added.\n");

}

**else** {

System.***out***.println("Appointment cannot be added.\n");

}

**if**(AppointmentService.addAppointment("D")) {

System.***out***.println("Appointment added.\n");

}

**else** {

System.***out***.println("Appointment cannot be added.\n");

}

//Add appointment twice to test

**if**(AppointmentService.addAppointment("C")) {

System.***out***.println("Appointment added.\n");

}

(Main class AppointmentService where addAppointment is defined)

**public** **void** addAppointment(String appointmentID, Date appointmentDate, String description) {

**if**(!getAppointmentList().containsKey(appointmentID)) {

getAppointmentList().put(appointmentID, **new** Appointment(appointmentID, appointmentDate, description));

}

**else** {

System.***out***.print("Error! Appointment ID: " + appointmentID + " already exist.");

}

}

Errors like this were very common and I could not figure out how to fix them.

* + 1. **Assess the ways you tried to limit bias in your review of the code. On the software developer side, can you imagine that bias would be a concern if you were responsible for testing your own code? Provide specific examples to illustrate your claims.**

I don’t have any kind of bias towards code that I’ve written or someone else. Errors are a problem no matter what form or who made them.

* + 1. **Finally, evaluate the importance of being disciplined in your commitment to quality as a software engineering professional. Why is it important not to cut corners when it comes to writing or testing code? How do you plan to avoid technical debt as a practitioner in the field? Provide specific examples to illustrate your claims.**

Quality work is what makes a programmer great, but quality work is not achieved alone. Writing code alone and editing it is an incredibly tedious process. Having a second set of eyes to catch errors or to point out points of efficiently is incredibly helpful and is the most realistic when it comes to working on projects such as this. Working in teams on projects like this make the process easier and more efficient and I plan to utilize team members in the future when it comes to coding. It may seem easy to cut corners when it comes to coding but all it does is cause more issued to arise down the line. Practicing tried and true testing methods such as JUnit test or dependency test is the most efficient way to code and the most efficient.

**Work Cited**

Graphics Systems and Methods for Accelerating Synchronization Using Fine Grain Dependency Check and Scheduling Optimizations Based on Available Shared Memory Space. (2021).