**Project Two**

Iesha Sahin

Southern New Hampshire University

CS 320: Software Test Automation& QA

Miguel A. Ayala

August 20, 2023

**Summary**

*To what extent was your approach aligned to the software requirements?*

My approach aligned with the software requirements because I divided all tasks into small digestible blocks to make my test cases. Because of this, my code met the requirements that were asked of me. An example would be in Appointment.java where past dates were not allowed to be requesting when making a new appointment.

private final Date date;

if (date == null || date.before(new Date())) {  
 throw new IllegalArgumentException("invalid date");

this.date = date;

public Date getDate() {  
 return date;  
}

@Test  
void pastDate() {  
 Date now = new Date();  
 Date past = new Date(now.getTime() - 1);  
 Assertions.assertThrows(IllegalArgumentException.class, () -> {  
 new Appointment("1234", past, "a description");  
 });  
}

Focusing only on the date, once declared I was able to make sure that only the current date and future dates were valid.

*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?*

I know that my JUnit tests were effective based on the coverage percentage because my percentage was over 80% which was the requirement of the project. I tested each prompt separately to ensure no errors, and then again to ensure it worked as a whole.

*How did you ensure that your code was technically sound?*

I made sure my code was technically sound by taking it one step at a time. I’ll use ID in Contact.java as my example.

public Contact(String ID, String firstName, String lastName, String phone, String address) {  
 if (ID == null || ID.length() > 10) {  
 throw new IllegalArgumentException("ID invalid");  
 }  
 }  
  
 this.ID = ID;  
}

Once this was written, I took time to look for any spelling errors as I make them much more than I would like to. After I was positive everything was free of errors, it was time to test in ContactTest.java.

For the first test we’re looking to make sure that the user ID is no longer than 10 number long.

@Test  
void IDLength() {  
 Assertions.assertThrows(IllegalArgumentException.class, () -> {  
 new Contact("01234567890", "Jill", "Smith", "0123456789", "1 main st");  
 });  
}

Once and only once this passed, I was able to move on to my next ID test. I had to ensure that the ID field was not able to be left null.

@Test  
void idNull() {  
 Assertions.assertThrows(IllegalArgumentException.class, () -> {  
 new Contact(null, "Jill", "Smith", "0123456789", "1 main st");  
 });  
}

Everything compiling with no bugs told me that the code was technically sound, and I could move on to the next portion.

*How did you ensure that your code was efficient?*

I made sure my code was efficient for someone who might go in and look at my code when I’m not around is by grouping things together.

Here I’m making sure that both the getter and setter for lastName are together.

public String getLastName() {  
 return lastName;  
 }  
  
public void setLastName(String lastName) {  
 if(firstName == null || lastName.length() > 10) {  
 throw new IllegalArgumentException("Last name invalid");  
 }  
 this.lastName = lastName;  
}

Here I’m making sure that all tests for lastName are together.

@Test  
void lastNameLength() {  
 Assertions.assertThrows(IllegalArgumentException.class, () -> {  
 new Contact("123", "Jill", "SmithSmithSmith", "0123456789", "1 main st");  
 });  
}  
  
@Test  
void lastNameNull() {  
 Assertions.assertThrows(IllegalArgumentException.class, () -> {  
 new Contact("123", "Jill", null, "0123456789", "1 main st");  
 });  
}

I think it makes it easier for me if I ever need to go back and revise, but also easier for a fresh pair of eyes who might want to have a look too, as they won’t have to look in a million different places to find functions relating to the same thing.

**Reflection**

*What were the software testing techniques that you employed in this project?*

A software resting technique I used were unit tests. With unit tests, I was testing small blocks of the code to ensure they worked before moving on. By doing so, I was assured that all code was working as intended. The unit tests written were made to catch all possible defects one at a time before coming together. I think I also used integration testing, but I’m not entirely sure if I’m honest. Integration testing is when different components of the project are all tested as a whole. I say I think I used this technique because when implementing the getters and setters, they were declared in the object class but weren’t used until the service class. For them to have been used in the service class, the entire project had to be tested as a whole.

*What are the other software testing techniques that you did not use for this project?*

A technique that I did not use would be regression testing. Regression testing ensures no new bugs have shown up after a code update or change. Once all my test passed, I did not go back to update any code so there was no reason for this technique when it came to my project.

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

Unit testing is practical because it makes sure the code you are building is both error free and strong. Integration testing is great because once you have all your code and their tests working, and it’s time to put it together, you can test that all components are working with each other. Just because a block of code works on its own doesn’t mean it will work with the rest of the system, and integration testing ensures it will. Regression testing is important for any updates because, as I just mentioned, just because new code works on its own, you must confirm it works with the rest of the preexisting code and generates no errors.

**Mindset**

*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?*

For my project, I didn’t employ too much caution. I believe this to be the case because the prompts weren’t too intense. I did struggle at times, but it was tame compared to what a project in the real-world would be. Caution is very important though. Without caution as a software tester, there is a potential for huge consequences, and possibly even loss of money. For the sake of the project, it is important to appreciate the complexity of the code I was testing because it had a direct effect on the coverage of my tests. Like the example I used before with the getters and setters. The getters and setters were all declared in the object class but were not implemented. All getters and setters were implemented in the service class. This is one of the things that confused me with the test coverage because I couldn’t understand why the percentage was so low until I realized that once I finished with the service class, and they were being implemented, my percentage went up.

*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?*

I don’t know that I had any bias in my review of the code. Code itself is unbiased. I think as you become a more experience developer you gain a bit of an ego, or just more pride in your work. That’s when I might have to try and limit my bias. I know developers get attached to their code, but I think it’s important to understand that it’s not personal. When testing your own code, you just have to remember it’s not about you, but the end goal. Failed tests or code that could be written better is all okay and will only make you that much better.

*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?*

Discipline is imperative to have as a developer and professional in general. To produce quality results, cutting corners should be avoided at all costs. People can tell when corners are cut, in all aspects of life. One thing I have learned in the past eight weeks is: testing, testing, testing. Imagine I just didn’t feel like testing one day and submitted whatever I whipped up? Nothing would have passed. Cutting corners with your code only allows for horrible consequences like bugs, and even worse, security breaches. A lot of times too, taking the easy way out trying to save time and money will just cost you more time and money in the end. My plan to avoid technical debt is making sure I’m working with a team that cares and will hold each other responsible. If working alone, making sure that I remember the potential consequences of my actions. Also, staying proud of my work is a huge deal to me.

**Citations**

Awati, R. (2022). integration testing or integration and testing (I&T). *Software Quality*. https://www.techtarget.com/searchsoftwarequality/definition/integration-testing?Offer=abMeterCharCount\_var2

Katalon. (2023, August 3). What is regression testing? Definition, tools, examples. *katalon.com*. https://katalon.com/resources-center/blog/regression-testing