# Southern New Hampshire University

7-2 Project Two Submission

Anthony Spedaliere

Professor Kraya

CS-320-T1183 Software Test Automation and QA

October 10th, 2022

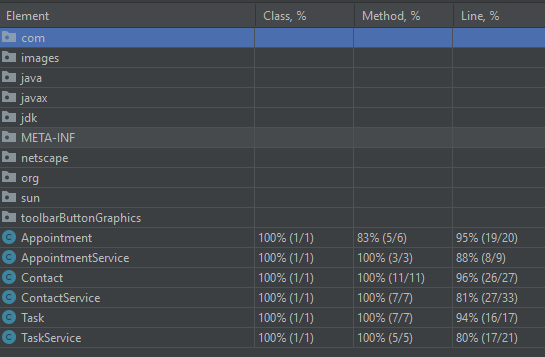
Project one consisted of six Java classes: Contact, ContactService, Task, TaskService, Appointment, and AppointmentService. Each class had a corresponding test file associated with it: ContactTest, ContactServiceTest, TaskTest, TaskServiceTest, AppointmentTest, and AppointmentServiceTest. Each class was outlined through requirements in the project description. For example, the Contact class contained five requirements. Here is requirement number two: “The contact object shall have a required firstName field that cannot be longer than 10 characters. The firstName field shall not be null.” These class requirements served as the specification for my tests and served to guide me in my testing approach. For example, here are some of the corresponding tests that are associated with the above requirement.

@Test  
void testIfFirstNameTooLong() {  
 *assertThrows*(IllegalArgumentException.class, () -> {  
 new Contact("1", "Tony The Iron Man", "Stark", "1234567", "111 Secret Hideout");  
 });  
}

@Test  
void testIfFirstNameNull() {  
 *assertThrows*(IllegalArgumentException.class, () -> {  
 new Contact("1", null, "Stark", "1234567", "111 Secret Hideout");  
 });  
}

The second requirement consisted of two main concepts. The first was that the firstName field was to not be longer than 10 characters. The second element was that it was to not be null. Rather than creating one test that encapsulates both concepts, the single requirement was separated into two tests. By separating the requirement into individual tests, I was able to ensure that my tests were more technically sound and more readable. Also, by focusing each test on only one specific element of the requirement it reduces the chance of a false positive.

It was also important to check that the total coverage of all the tests was sufficient. The project called for 80% coverage. I set up my project to provide me with a report on the total coverage based on total class, method and line covered by percentage. I provided a screenshot of this report below.



The most stringent coverage is the line percentage. I ensured that my tests covered at least 80% of the total lines contained in my code. This analysis helps to ensure that my tests are effectively reaching the majority of lines and most important portions of my codebase and gives concrete evidence to support the effectiveness of my tests. Test coverage alone is not sufficient. The tests must also cover the requirements.

The two main categories of software testing are static testing and dynamic testing. Static testing is the process of locating errors without executing the code and dynamic testing is the process of locating errors at runtime (GeeksforGeeks, 2021a). Unit tests are a form of dynamic testing (Sharma, 2022). The primary form of testing done during this project was unit testing. Typically, after unit testing integration testing, system testing and acceptance testing are performed, but some of these can be done concurrently or redone. Static testing can be performed by the code author or by the team. It is suggested that someone other than the author perform a review of the code in order to avoid bias.

Coder bias is important to consider when designing tests. The most biased individual in the testing process is the person or persons who originally authored the code. For this project I wrote both the code and tests. Ideally, another member of my development team, an independent test team, or an outsourced testing consultant would be responsible for writing the test code. The further removed from the original code the less biased the tests will be on average (Software Testing: An ISTQB-BCS Certified Tester Foundation Guide, 2015). I like this quote from our text, “Testing can be more effective if it is not undertaken by the individual(s) who wrote the code, for the simple reason that the creator of anything (whether it is software or a work of art) has a special relationship with the created object (Software Testing: An ISTQB-BCS Certified Tester Foundation Guide, 2015).” As someone who has undertaken many creative ventures and spent time with artists both giving and receiving critiques, this is true. There have been times when I author or create something with a glaring error, and I am completely blind to its existence until someone else points it out. In contrast, I have experienced the opposite, when I critique another, and I point out their glaring mistakes. There is a reason why the “editor” role exists in almost every industry (maybe by another title such as reviewer).

When designing tests for software it is important to be disciplined and not cut any corners. Software tests are incredibly important. They can be the difference between life and death in some cases. One example is the explosion of Ariane 5 rocket. According to an online article written by Douglas N. Arnold, a rocket that cost 7 billion dollars to develop was loaded with 500 million dollars of cargo. On its first flight it exploded, and all the cargo was destroyed. Fortunately, it was unmanned. The failure was due to an integer overflow when a 64-bit float was converted to a 16-bit signed integer (Arnold, 2000). This error could have been avoided with more extensive testing. This event occurred in 1996 so technology and testing theory were not as advanced. However, it serves as an example that a relatively small error can have enormous consequences. Ultimately, the more time you put into your tests downstream of the project will save time, money, and improve the quality of your product upstream (Software Testing: An ISTQB-BCS Certified Tester Foundation Guide, 2015). In extreme cases/projects, you may even be saving lives.

Throughout the course of this project, I have learned the pros of writing good tests and the cons of neglecting to write tests. Testing is a vital stage in the software development lifecycle. I look forward to learning more about this subject as I progress in my career as a software developer.

References

*Software Testing: An ISTQB-BCS Certified Tester Foundation guide*. (2015, June 18). BCS, The Chartered Institute for IT.

Arnold, D. (2000, August 23). *The Explosion of* *the Ariane 5*. University of Minnesota. Retrieved October 2, 2022, from https://www-users.cse.umn.edu/%7Earnold/disasters/ariane.html

GeeksforGeeks. (2021a, March 1). *Software Testing Techniques*. Retrieved October 2, 2022, from https://www.geeksforgeeks.org/software-testing-techniques/

Sharma, L. (2022, October 2). *Difference between Static Testing and Dynamic Testing*. TOOLSQA. Retrieved October 2, 2022, from https://www.toolsqa.com/software-testing/static-testing-and-dynamic-testing/#:%7E:text=Unit%20tests%2C%20integration%20tests%2C%20system%20tests%2C%20and%20acceptance,flaws.%20Difference%20between%20Static%20Testing%20And%20Dynamic%20Testing