Martin Salas

CS 320

Dr. Bolton

October 16, 2022

Project Two

My approach was in strict accordance with the requirements because I prioritized creating code with reliable tests that met all of the criteria. Consider the contact class. According to the requirements, both the first and the last name must be present and cannot be null. As a result, the character value must include no more than ten individuals at most. In the JUnit test for our ContactTest class, there was an assertion that checked to see whether the input was too lengthy. The maximum length of the task ID/unique ID was capped at 10 characters, as specified by the task class. In order to accommodate a task ID that may include more than 10 characters, the TaskTestJUnit test that we developed made use of a method that threw an argument exception.

The resources included in each module made a significant contribution toward increasing the coverage of code that was tested, which, in turn, led to an overall increase in the quality of the JUnit tests that I created week after week. I was able to have complete or very near to complete coverage across all three milestones, when all coverages were compared to one another. I knew that the JUnit tests I was running had successfully covered the majority of the code's functions whenever a positive coverage percentage was shown since these tests were being performed by me.

To ensure that my code met all of the necessary technical standards, I employed a HashMap. To give you an illustration, you may see me utilizing it in any one of the three milestones. I made use of HashMap in order to maintain track of the contacts that each had a distinct id. For HashMaps, it is a key/value pair, and there can only be one unique key for each value. This made it much simpler for me to verify for duplication, and it also sped up the process of locating a certain task, appointment, or contact using the key id. Additionally, the HashMap made it simpler to delete contacts, tasks, and appointments as well as create new ones.

I ensured the functionality of my code by doing a number of tests. Before I started this class, I did have some experience with coding, and that experience was really helpful when deciding which of the different data structures to use. I attempted to validate all of the variables in my Contact and Task classes using a single method in order to cut down on the amount of code that I had to write and to eliminate any redundant code. Last but not least, while creating variables for any of my classes, I exercised extreme care and set the majority of them to private status so that other classes could not access them.

Testing is an essential component to the success of any project. When it came to employing software testing approaches for this project, I used a plan that was quite precise. As we went through the criteria, I had the impression that I had a tested for everything, such as the maximum number of characters that both a first and last name may have and the fact that neither name can be null. When I was naming my tests for the JUnit tests, I made sure that they were named in a manner that was pretty close to the names of the main classes. Because of this, everything was maintained in a very orderly manner. One further technique for testing software that I used was called the step-by-step approach. Where I went over each criteria one at a time to make sure I didn't leave anything out and ensure that I didn't forget anything. After ensuring that all of the requirements had been satisfied, I went ahead and created my test cases (JUnit). In addition to JUnit testing, I made use of the White Box Testing methodology in order to complete any structural and logical testing that needed to be done.

I did not put any effort into using the approach of software testing known as static testing for this project. Examining source code without actually executing the program in order to locate bugs is what's meant by "static testing." My code is never only looked through; it is always executed as well. On the other hand, one of my favorite things to do is to hit the Build All or Debug button and wait for issues to occur. I am aware that the objective of static testing is to identify problems as early as possible in order to implement preventative measures. This is accomplished via walkthroughs, code inspections, and peer reviews.

Since I first began working on this project, I have experienced a profound shift in perspective. To tell you the truth, before to taking this class, I had very little experience with any form of software testing at all. In all honesty, I was under the impression that "testing" meant just executing the application in order to look for any issues that may occur. Being a software tester requires more commitment than caution, in my opinion. I would describe it as committed work. We are not allowed to take any shortcuts in order to finish a program more quickly, and since it is only natural for us to want to ensure that the final product satisfies all of the client's requirements, we are obligated to make sure that the quality and functionality of the final product are not diminished in any way.

If you created and tested your own code, I can see why someone could feel there is a bias in the results. Maintaining a humble attitude and being open to new information and experiences is, in my view, the easiest approach to reduce the impact of prejudice on one's thinking. If we test our own code on the false assumption that we are the finest programmer in the world, we will never learn anything new or discover any bugs. Because of this, our finished product might end up being lower in quality. My commitment to excellence as a professional in the field of software engineering requires me to maintain a high level of self-discipline. Not only do we abide by a stringent set of software engineering principles, but we also make it quite clear that cutting corners on the development and testing of code is never okay.