Andrej Oljaca

Project Two

SNHU

Project 2

Describe your unit testing approach for each of the three features.

My unit test approach for each of the three features was similar. I created tests to test all of the functions that I created. I made sure the create tests using JUnit. Also I made sure that all the packages and imports were there.

* To what extent was your approach aligned to the software requirements? Support your claims with specific evidence.
  + I made sure that my unit testing aligned with software requirements. I did this by testing all the functions that were created and making sure that the green bar appeared. I also made sure that the test coverage was above 80%.
* 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 tests were effective because I tested each function and made sure that they performed as expected.

Describe your experience writing the JUnit tests.

* How did you ensure that your code was technically sound? Cite specific lines of code from your tests to illustrate.
  + I made sure that my code was technically sound by using encapsulation, and breaking out functions into their own functions, when possible, to ensure easier reading and understanding of the code. I also made sure to provide robust comments. I also used correct indentation and syntax such as semicolons at the end of each line/statement. Encapsulation can be seen at the beginning of my code where I define classes. Another important thing I did to make my code technically sound was review it and double check it. I also made sure to resolve all errors that appeared.
* How did you ensure that your code was efficient? Cite specific lines of code from your tests to illustrate.
  + I made my code efficient by using the correct data structures such as arrays when I needed to hold data. Also I made sure to not make any unnecessary calls or operations. I also used functions to break my code into readable and efficient code. On top of that, I used packages when I could to ensure that I didn’t need to write any unnecessary code. Also I avoided using unnecessary variables. I used loops when possible and broke out of them when possible.
* What were the software testing techniques that you employed in this project? Describe their characteristics using specific details.
  + One of the static testing techniques I employed in this project was static testing. Before I executed my code I reviewed it and made sure that there were no errors and that my code was technically sound and efficient. I also dynamically tested my code using JUnit. I made sure that the functions that I wrote worked as expected.
* What are the other software testing techniques that you did not use for this project? Describe their characteristics using specific details.
  + One testing technique that I did not use for this project was the use of decision tables. Another was use case-based testing which describes the interaction between the actor and the system. I also did not use equivalence partitioning, error guessing, or exploratory testing.
* For each of the techniques you discussed, explain the practical uses and implications for different software development projects and situations.
  + Each of these testing techniques has different practical uses and implications. Decision tables are good for testing decisions made depending on given conditions. Use case-based testing helps to test the entirety of the system. Equivalence partitioning divides the input data of a software unit into partitions of equivalent data from which test cases can be derived. Error guessing is a technique where the tester tries to guess what will cause errors. Exploratory testing focuses on discovery and relies on the guidance of the individual tester to uncover defects that are not easily covered in the scope of other tests.
* 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.
  + I employed caution to a certain degree. The good thing about Eclipse is that it shows you errors in your code along with solutions to the errors. For example when I needed to access models from folders outside of the current folder Eclipse showed me how to do that. Or when I needed to make certain imports or define packages, Eclipse did this for me. On top of this it helped me with syntax errors. It also helped me export my code and test it.
* 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 can imagine that bias can be a problem when reviewing your own code. Since you wrote it you might think that it is correct even when it is not. On top of this, it is hard to see your code from a fresh perspective which is necessary as part of the review process. When you are coding you are in the weeds and don’t have a bird's eye view, but the reviewer should precisely have that.
* 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.
  + It is important to not cut corners when writing or testing code, because code is very nitpicky. Just one mistake can lead to bugs and the crashing of the program. On top of this, the business might rely on the software for profit and cannot afford any app crashing bugs. To add to that, when you need to keep adding to your code, your code needs to be architected correctly. Others should be able to read your code easily and review it, which can be done through the use of good variable and function names.