Vincent Snow

April 10 2023

Project two

The unit tests written for the software address each of the specifications given in the software requirements: for example, confirming that entering a description with over 50 characters will be rejected and will result in an error message. While coding, I stepped through a checklist of each one of these requirements and coded them out while checking for typing errors with the IDE's built-in corrections. Then, I stepped through each of the methods of the code and wrote a test for them, and one to confirm each of the requirements by calling the method with invalid input. The effectiveness of the tests is reflected in the percentage of coverage, which is over 80% for each class test. This is considered effective coverage and reflects how many lines of the code were confirmed to function properly through individual tests. The code was made technically sound by following Java standards such as proper class formatting with encapsulated private data fields and abstracted methods to allow viewing or changing the data rather than direct access. I made sure not to use any deprecated methods and paid attention to warnings in the IDE while working, which helped me to prevent other problems such as unused variables or imports. The coding was done efficiently by using variables to replace repetitive lines of code, such as in the test cases where sample objects for testing are created once and stored in a variable that is then used in each test algorithm. This cuts down on data usage and time to run the program while also making the code more readable and comprehensive.

The tests used in this project are a type of automated testing called unit testing. These tests check that individual methods and classes are working properly through execution of a pre-written

script. In this case, it is also a form of functional and acceptance testing – these tests were written to ensure that the business requirements given were met, and that specific data was returned as expected when a method was executed.

Manual testing methods were not used here on actual running code, which could have been done by using a temporary printed menu for input to check that each part of the code was working properly. There was no performance or integration testing necessary in this phase, since no other parts of the code such as databases are involved, and the program is in its beginning bare-bones stage.

Manual testing can be useful for testing a more developed, working code to find weaknesses or bugs. Automated test scripts are more useful for functionality or checking for requirements since they are very quickly run and will alert the developer immediately when a change to the code has caused a problem. Performance testing would be important to do once the program is running on its chosen server to be sure it can meet performance requirements such as handling many users at once. The functional unit tests should be scripted alongside each new part of the code to make sure that it works correctly, saving time on errors and mistypes as development continues.

While writing the tests, I kept a cautious and thorough mindset. Each line of the code could potentially be affected by the lines preceding it. The tests were written in order to make certain that even the class initializing functions were working as expected, which would be a pre-requisite for any other function within the class to work properly. Thorough testing like this avoids bias by testing every part of the code even when I feel very strongly that it will work because of previous experience, since it is possible that I could have missed a detail. This overconfidence bias, an assumption that your code is correct, is one of many types of bias a developer should be aware of (Cneude, 2021). Being disciplined in your testing enough to write

detailed tests like this is important because small mistakes that are overlooked can have disastrous results if the flaws are published and the software is in use by customers. It could cost the developer or company large amounts of money or have legal repercussions. It could result in people going through hardship due to their personal details being revealed to criminals. In certain industries, such as in the operation of aircrafts or medical equipment, it could even cost people their lives.

References

Cneude, M. (2021, March 17). *8 Cognitive Biases in Software Development*. The Valuable Dev. https://thevaluable.dev/cognitive-bias-software-development/