**Unit Test and Refactoring Questions**

by Sujan Kafle

s315570

PRT452 Software Engineering: Process and Tools

Tutor: Dr Charles Yeo

Charles Darwin University

15th September 2019

**Program to create a number guessing game:**

TDD Approach for method *playOrQuit*:

|  |
| --- |
|  |
| Method playOrQuit is created. This is the skeleton to check weather user’s input matches the required input. |
|  |
| This is the Junit test code for method playOrQuit. It should display test failure for the first time. |
|  |
| For the first time the test fails. |
|  |
| String “WWW” is taken as input which is false. The test result should be failed for this. |
|  |
| It is failed because the method is created to accept the user input only “Q” or “S”. |
|  |
| User’s input is taken as “Q”, it should pass the test result. |
|  |
| It shows the passed test results. |
|  |
| Similarly, the program should accepts the user’s input as “S”. It should pass the test result. |
|  |
| Finally, the block of code (playOrQuit) for user’s input to start or quit the game has been passed successfully. |

2. TDD for method *numberOrNot*:

|  |
| --- |
|  |
| This method checks whether the given user’s input is an integer or string. It will return true value for the integer number and false for the string one. |
|  |
| This is the test code skeleton for this method to examine the given value is integer or string. |
|  |
| Here, the given value is “TEST” (string) so it should fail the test result. |
|  |
| Since, the user’s input is String the test is failed. |
|  |
| Now, the user’s input is made integer i.e. ‘20’ and it should pass the test result. |
|  |
| Finally, it shows the tests passed. |

3. TDD Approach for *greaterOrNot*:

|  |
| --- |
|  |
| *greaterOrNot* methods evaluate the given user input which should not be greater than 100 and lesser than 0. |
|  |
| “200” value is taken as user input which violates the program syntax, it exceeds 100. Thus, the test should fail. |
|  |
| The test is failed. Here, the user is prompted to enter the value between 1 to 100. |
|  |
| Now, the input value is “20” which is less than 100 and greater than 0. So, it should pass the result. |
|  |
| Finally, the test passed. |

4. TDD Approach for ***matchWithRandomNum***:

|  |
| --- |
|  |
| This function works as comparing user input with the random number so that user will be notified that their guessed number is matched or not, and it will keep asking until it is matched or has quit the game. |
|  |
| Test code for the method *matchWithRandomNum* where the given number (80) is more than random number (50). It should fail the test result. |
|  |
| Finally, it is failed due to the unmatched number with random number. |
|  |
| Similarly, the user input is less than the random number. It should fail the test. |
|  |
| The test fails since the user input value is less than random number. |
|  |
| Here, the user input is equal to the random number. It should pass the result. |
|  |
| Finally, it works. Test passed!! |

5. TDD Approach for whole program!!

|  |
| --- |
|  |
|  |
|  |
| This method will apply conditions for the user’s input value where “Q” implies for quitting the game and “S” to start the game. If no conditions are matched it will prompt the users to enter valid input to play or quit the game. |
|  |
| Firstly, it calls the function *numberOrNot* method to check whether the given input is integer or not. Secondly, it helps to validate the user’s input whether it is greater than 100 or less than 0. Then, it calls the function *matchWithRandomNum* to match the input with random number. |
|  |
| This code snippets checks the given user’s input is either integer or string. |
|  |
| Similarly, this method has different conditions where user’s input is compared until it matches the random number. |

6. Final Screenshots of the game:

|  |
| --- |
|  |

7. Refactored Code: Final Code

|  |
| --- |
|  |
| All of the required variables are made global variables to be accessible easily. |
|  |
|  |
|  |
| Few duplicated codes are analyzed and made concrete structure by eliminating those codes. |
|  |
|  |
|  |

Git Hub Link:

<https://github.com/sujankafle/PRT452GuessingGame>

# References

Sourcemaking.com. (n.d.). *Design Patterns and Refactoring*. [online] Available at: https://sourcemaking.com/refactoring/smells [Accessed 16 Sep. 2019].

Farcic, V. (2013). *Test Driven Development (TDD): Example Walkthrough*. [online] Technology Conversations. Available at: https://technologyconversations.com/2013/12/20/test-driven-development-tdd-example-walkthrough/ [Accessed 16 Sep. 2019].

Pragati Software. (n.d.). *Test-Driven Development: An Introduction with a JUnit Example - Pragati Software*. [online] Available at: http://pragatisoftware.com/test-driven-development-an-introduction-with-a-junit-example/ [Accessed 16 Sep. 2019].

luv2code (2014) *Java Eclipse Tutorial - Part 9: Running JUnit Tests.*Available at: https://www.youtube.com/watch?v=ouLxs8UHxjg&feature=youtu.be&list=PLEAQNNR8IlB5R\_qZvrFH0g8wC\_FLWK9si [Accessed 16 Sep. 2019].