Software Development- 1DV600

Assignment 3

Project Hangman

Name: Nitin Pasikanti

Lnu-email address:
np222@student.lnu.se

Link to GitHub repo:

https://github.com/np222gb/np222gb_1DV600/tree/master/Assign%203

CONTENTS

Vision

The Hangman project is based on the word guessing game called ‘Hangman’. In this game, players try to figure out an unknown word by guessing letters. The game begins with a greeting message, ‘Welcome to the game of Hangman. Save the man, Win the game!!’. The user follows the instructions on the screen. When the game begins, the player has limited number of chances for guessing. If too many letters which do not appear in the word are gussed, the man is hanged (and loses).

Use-cases:

This is a simple application with two use-cases.

UC 1 Enter a username

Pre-condition: Start the game

Post-condition: A message is shown to enter the player name and it is stored.

Main scenario

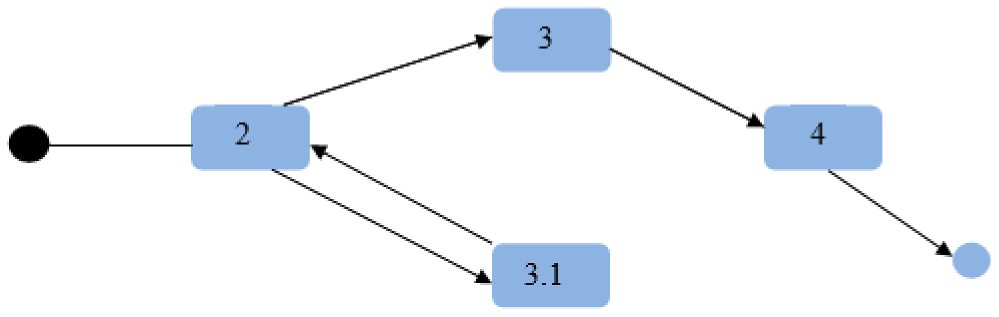
1. Starts when the user wants to store their name in the hangman game secession.
2. The system asks for a name.
3. The user provides the name.
4. The system stores the name and asks to choose the level of difficulty.

Alternative scenarios

3.1 The player provides an empty name.

1. The system shows an error message.
2. Go to Step 2 in Main Scenario.

Activity Diagram



UC 2 Select Difficulty

Pre-Condition: Enter your name

Post Condition: Difficulty Menu is shown

Main Scenario:

1. Starts when the user has given a name.

2. The system presents the difficulty menu, which displays the level of difficulty with three options:

A → Child (10 chances), B→ Adult (7 lives), C → Expert (5 lives)

3. The player selects the difficulty from the three options.

4. The system then starts the game on the desired difficulty level.

Alternate Scenarios:

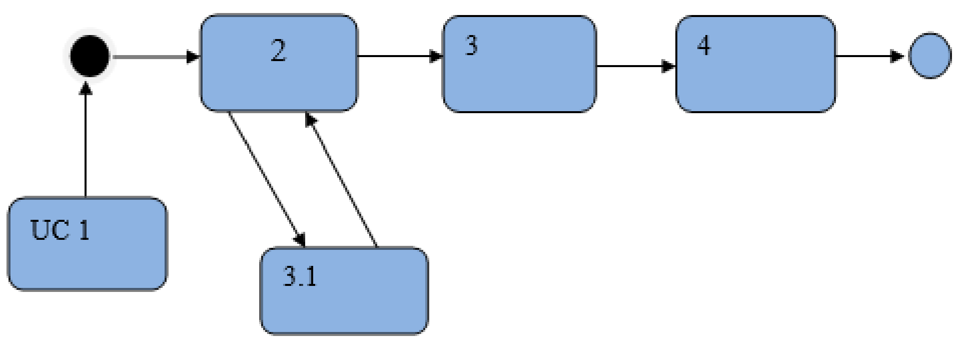
1. The user has not stored a name

    1. Go to ”UC 1 Enter a username”

3.1 Invalid menu choice

    1. The player makes an invalid choice by selecting a letter other than a, b or c.
    2. The system presents an error message.
    3. Go to Step 2 in Main Scenario.

Activity Diagram



Test Plan

- Objective of Testing in this interation

- The main objective is to learn the idea of creating a test plan and examining the code by static and dynamic inspection.
- This is an important tool for the project as it gives a detailed description of the features that need to be added or tested and finally the time spent in finishing the project.

- In this project of Hangman we are going to test mainly two use-cases by writing and running dynamic manual test-cases for each of the use- case:
  - UC 1: Enter a username
  - UC 2: Select Difficulty

- We also write automated unit tests for the PlayerName.java- methods in the class PlayerName_Test and for the DifficultyLevel.java in DifficultyLevel_Test.

- I did not test the setters and getters methods because the IDE can generate these methods, so I don’t think that will be a bug with generating it.

- I have tested these methods because they are important for the game to run accurately. Any errors in these methods would not make the game work as it should be.

Time Plan

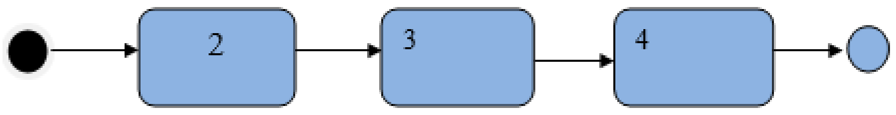
Task	Estimated Time	Actual Time
Manual TC	2 h	3 h
Unit Tests	2 h	4 h
Running manual tests	15 m	5 min
Code inspection	1 h	40 min
Test Report	3 h	4 hrs

Manuel Test- Cases:

TC1. 1 User enters a valid input

Use-case: UC1 store a Name.

Scenario: Check user input successful.



The main scenario of UC1 is tested where a user makes a valid input.

Precondition: Start the game.

Test Steps

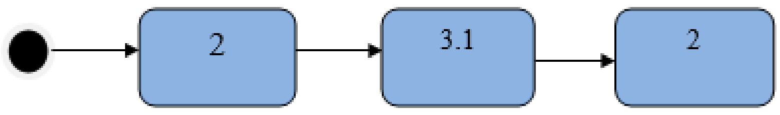
- Start the Hangman game
- The system greets the user and presents with 3 options.
- Choose option 1 → Play Game.
- The system shows the message ’Enter a username: ’
- Enter the name ’’nitin” and press enter.

Expected

- The system should proceed to the next step by prompting the user to select the difficuly level.
- Refer to the UC2.

TC1. 2 Store empty name force reprompt

Use case: UC1 store a Name.
Scenario: Store empty name force reprompt. The alterative scenerio where the user enters an empty username and is forced to enter a new username.



The alternative scenario of UC1 is tested where the user gives an invalid input.

Precondition: The user should have selected the option 1→ Start Game.

Test Steps
- Start the Hangman game
- The system greets the user and presents 3 options.
- Choose Option 1→ Start game
- The system shows the message ’Enter a username: ’
- Press enter without entering a name.

Expected

- The system should show the text " A username must have at least one character, try again"
- System shows "Enter a username: " and waits for input.

Test Report – TC 1.1, TC 1.2

Test traceability matrix and success

Test	UC1	UC2
TC1.1	1/OK	0
TC1.2	1/OK	0
COVERAGE & SUCCESS	2/OK	0

Automated unit test coverage and success

Test	PlayerName_Test	COVERAGE & SUCCESS
DifficultyLevel	0	0/NA
DifficultyLevel_Test	0	0/NA
Guessing	0	0/NA
Hangman_Main	0	0/NA
Instructions	0	0/NA
PlayerName	100%/OK	100%/OK

Comment

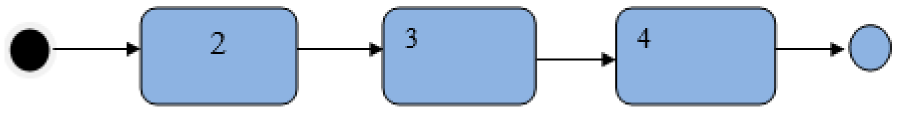
All tests pass, the table above has a record of the test traceability. The code runs successfully for a valid input and it displays an error message for an invalid input. Next iteration we need to focus on UC2.

The above test case is for storing a name which is contained in the class PlayerName.java. I have examined the code and most classes require mocks to be testable.

TC2. 1 Select Difficulty succcessful

Use Case: UC2 Select the level of Difficulty

Scenario: Difficulty selection successful



The main Scanario of UC2 is tested where the system asks the user to select the difficulty before starting the game. The system starts the game on the desired difficulty level.

Pre-condition: The user should have entered a name

Test Steps:

- Start the Hangman game
- The system greets the user and presents with 3 options.
- Choose option 1 → Play Game and hit enter
- Fill in a username. Eg: 'Nitin' and hit enter
- The system shows the following message, "Nitin before you start the game, please select the Difficulty Level
  A ---> Child (10 chances)
  B ---> Adult (7 chances)
  C ---> Expert (5 chances)
  Enter your choice [A,B or C]:

- Enter either 'b' or 'B' to choose the Adult difficulty level and press enter

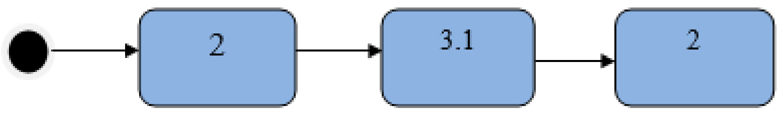
Expected

- The system should prompt to start the game and wish good luck.
- The 'Life remaining' should be 7

TC2. 2 Select Difficulty force reprompt

Use case: UC2 Select the level of Difficulty

Scenario: Enter an incorrect letter, force reprompt. The alternate scenario where the user enters an incorrect letter. An error message is shown and is forced to try again.



Precondition: The user should have entered his name in the UC 1

Test Steps:

- Start the Hangman game
- The system greets the user and presents with 3 options.
- Choose option 1 → Play Game and hit enter
- Fill in a name eg: ’Nitin’ and hit enter
- The system shows the following message, ”Nitin before you start the game, please select the Difficulty Level
  A ---> Child (10 chances)
  B ---> Adult (7 chances)
  C ---> Expert (5 chances)
  Enter your choice [A,B or C]:

- Enter ’e’ and and hit enter

Expected

- The system should show the error message ” Invalid input! Try again”
- System shows ” Enter your choice [A,B or C]: ” and waits for input.

Test Report – TC 2.1, TC 2.2

Test traceability matrix and success

Test	UC1	UC2
TC2.1	0	1/OK
TC2.2	0	1/OK
COVERAGE & SUCCESS	0	2/OK

Automated unit test coverage and success

Test	DifficultyLevel_Test	COVERAGE & SUCCESS
PlayerName	0	0/NA
PlayerNamel_Test	0	0/NA
Guessing	0	0/NA
Hangman_Main	0	0/NA
Instructions	0	0/NA
DifficultyLevel	100%/OK	100%/OK

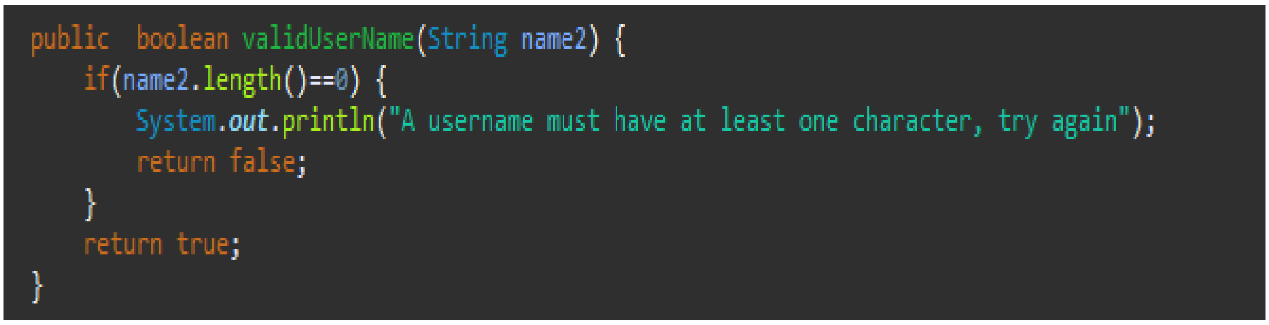
Comment

All tests pass, the table above has a record of the test traceability. The code runs successfully for a valid input and it displays an error message for an invalid input.

The above test case was for choosing the difficulty which is contained in the class DifficultyLevel.java. I have examined the code and most classes require mocks to be testable.

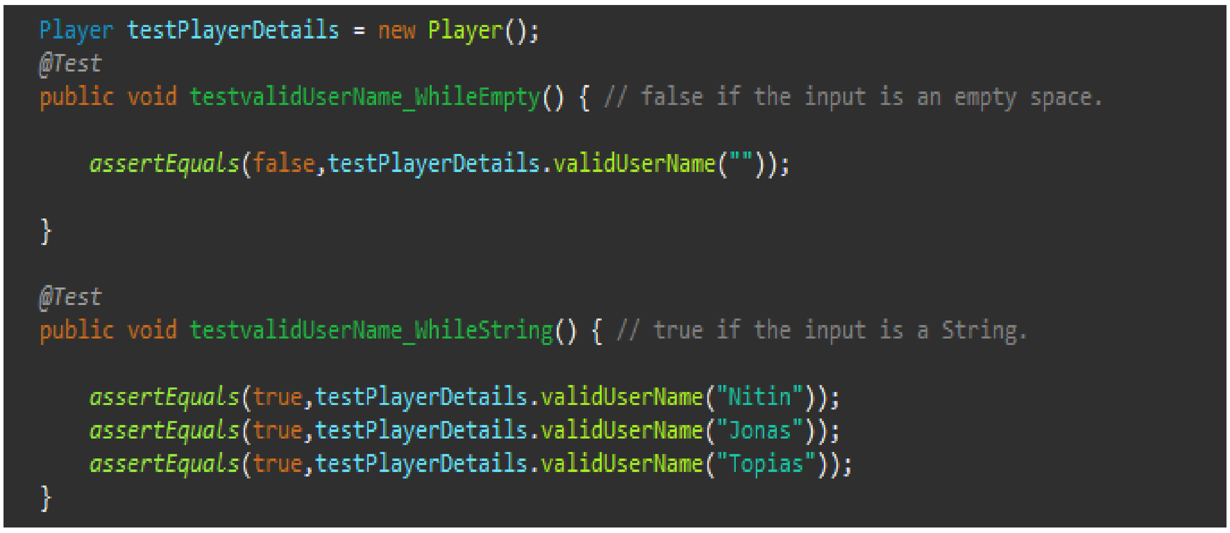
Unit Test

Source code: 1



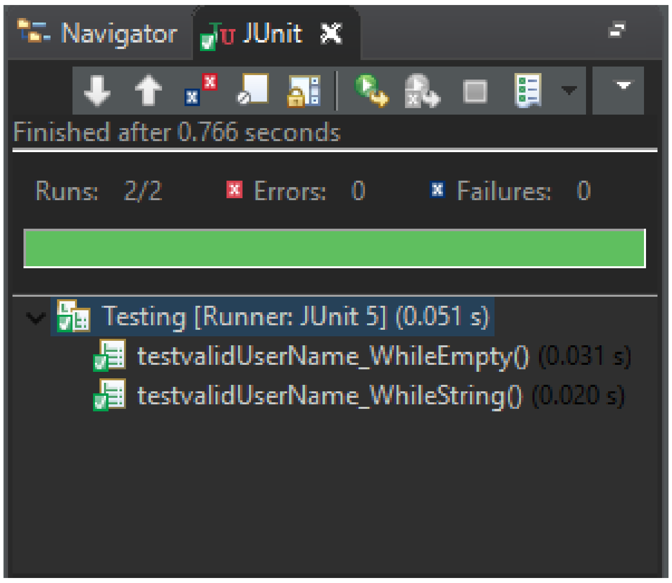
```
public  boolean validUserName(String name2) {
    if(name2.length()==0) {
        System.out.println("A username must have at least one character, try again");
        return false;
    }
    return true;
}
```

Junit test for S.code 1:



```
Player testPlayerDetails = new Player();
@Test
public void testvalidUserName_WhileEmpty() { // false if the input is an empty space.

    assertEquals(false,testPlayerDetails.validUserName(""));

}

@Test
public void testvalidUserName_WhileString() { // true if the input is a String.

    assertEquals(true,testPlayerDetails.validUserName("Nitin"));
    assertEquals(true,testPlayerDetails.validUserName("Jonas"));
    assertEquals(true,testPlayerDetails.validUserName("Topias"));
}
```

Result of testing S.code 1:



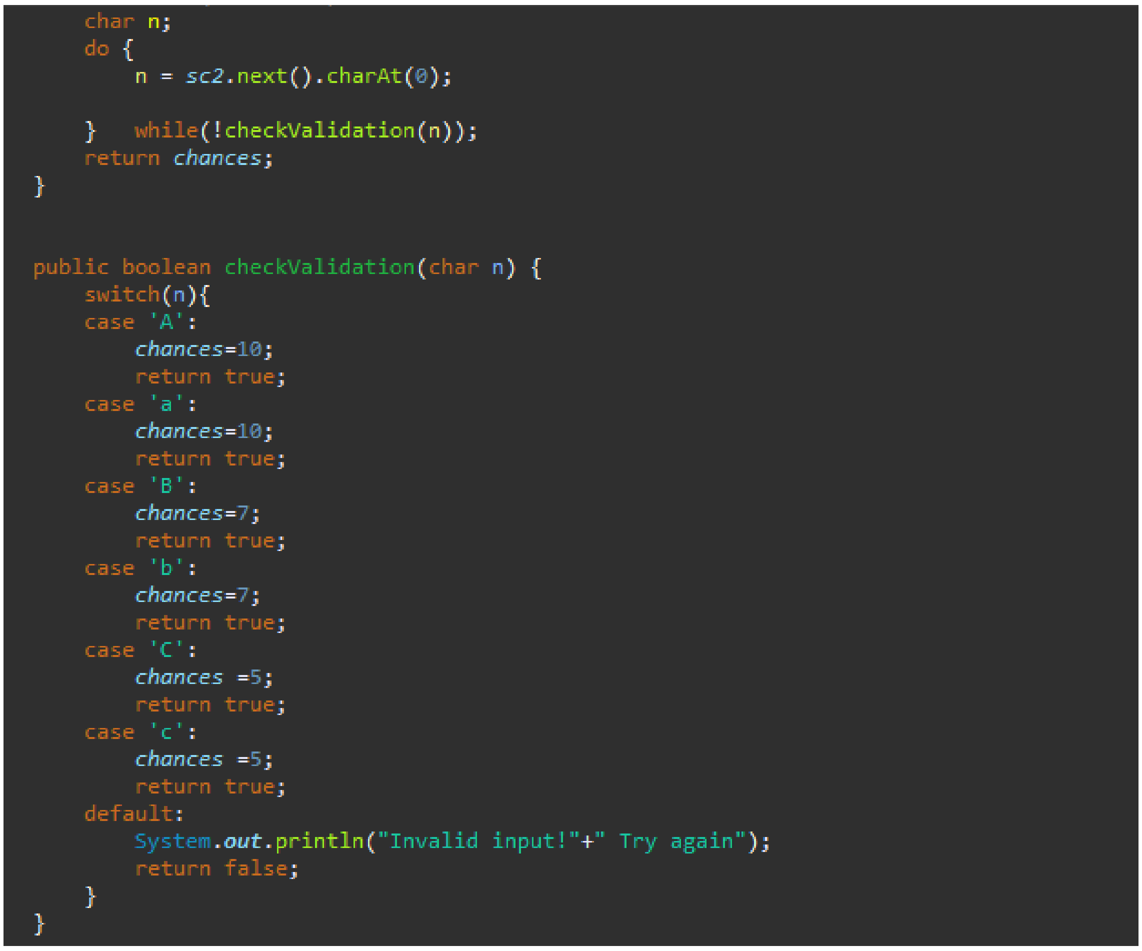
Navigator JUnit

Finished after 0.766 seconds

Runs: 2/2   Errors: 0   Failures: 0

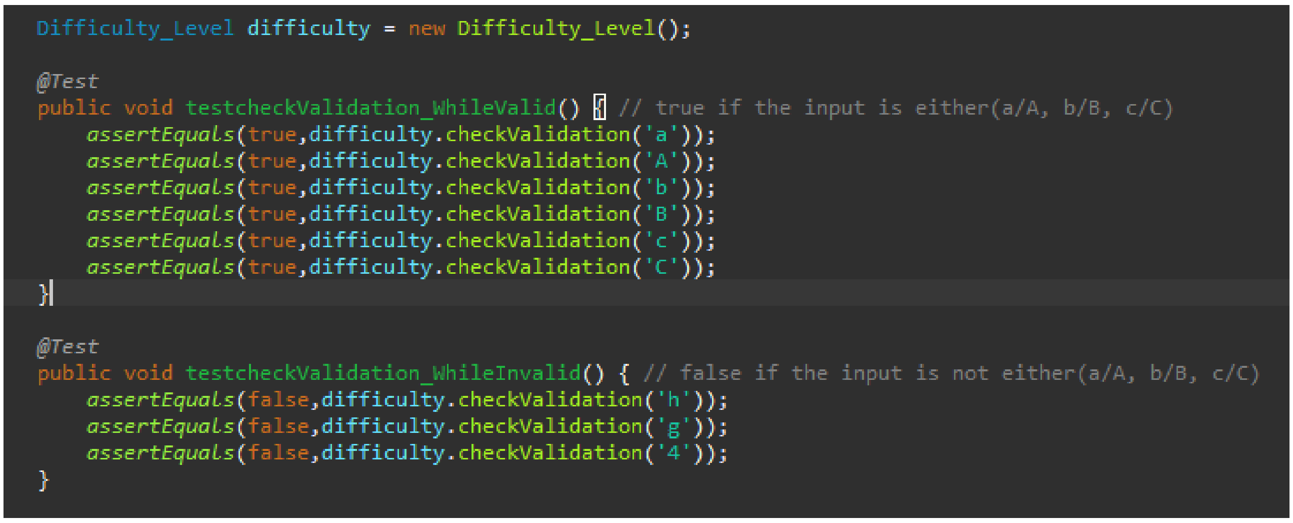
```
Testing [Runner: JUnit 5] (0.051 s)
    testvalidUserName_WhileEmpty() (0.031 s)
    testvalidUserName_WhileString() (0.020 s)
```

Source code: 2



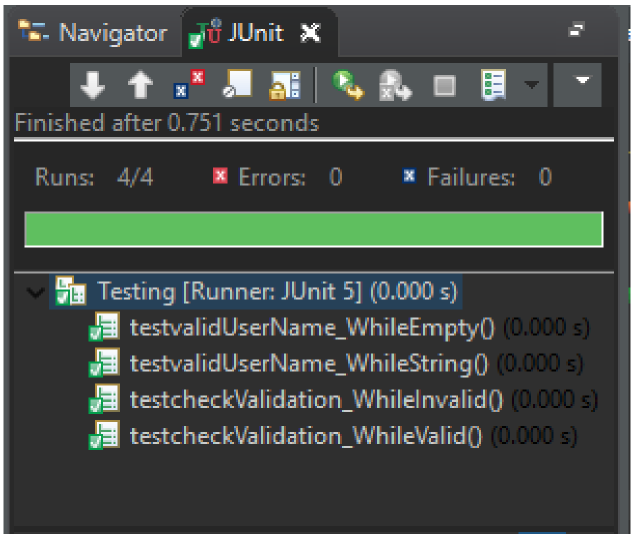
```java
    char n;
    do {
        n = sc2.next().charAt(0);

    }   while(!checkValidation(n));
    return chances;
}


public boolean checkValidation(char n) {
    switch(n){
    case 'A':
        chances=10;
        return true;
    case 'a':
        chances=10;
        return true;
    case 'B':
        chances=7;
        return true;
    case 'b':
        chances=7;
        return true;
    case 'C':
        chances =5;
        return true;
    case 'c':
        chances =5;
        return true;
    default:
        System.out.println("Invalid input!"+" Try again");
        return false;
    }
}
```

Junit test for S.code 2:



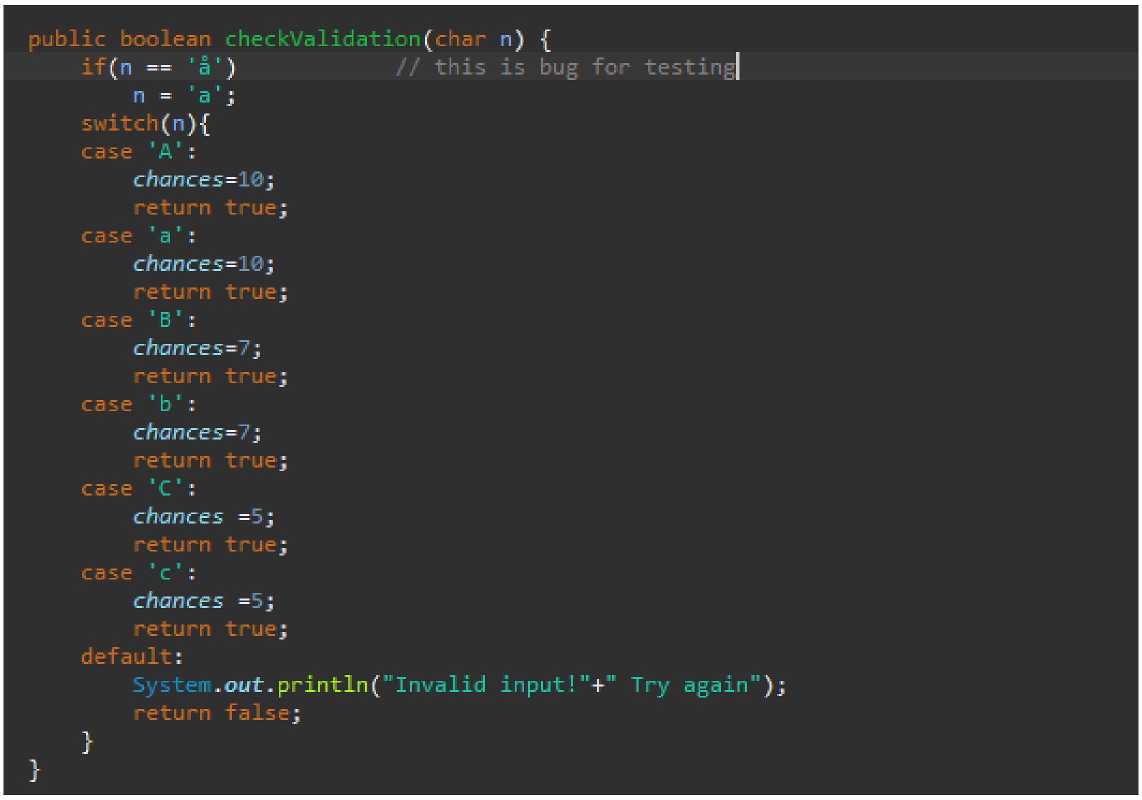
```
Difficulty_Level difficulty = new Difficulty_Level();

@Test
public void testcheckValidation_WhileValid() { // true if the input is either(a/A, b/B, c/C)
    assertEquals(true,difficulty.checkValidation('a'));
    assertEquals(true,difficulty.checkValidation('A'));
    assertEquals(true,difficulty.checkValidation('b'));
    assertEquals(true,difficulty.checkValidation('B'));
    assertEquals(true,difficulty.checkValidation('c'));
    assertEquals(true,difficulty.checkValidation('C'));
}

@Test
public void testcheckValidation_WhileInvalid() { // false if the input is not either(a/A, b/B, c/C)
    assertEquals(false,difficulty.checkValidation('h'));
    assertEquals(false,difficulty.checkValidation('g'));
    assertEquals(false,difficulty.checkValidation('4'));
}
```

Result of testing S.code 2:



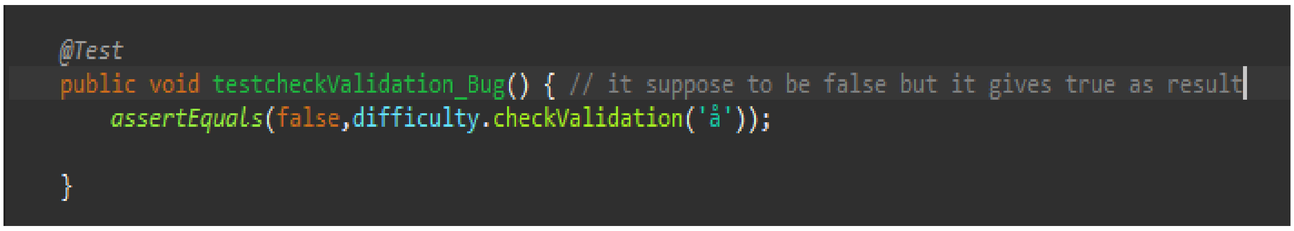
```
Finished after 0.751 seconds

Runs: 4/4    Errors: 0    Failures: 0

Testing [Runner: JUnit 5] (0.000 s)
    testvalidUserName_WhileEmpty() (0.000 s)
    testvalidUserName_WhileString() (0.000 s)
    testcheckValidation_WhileInvalid() (0.000 s)
    testcheckValidation_WhileValid() (0.000 s)
```

Source code 3: Bug



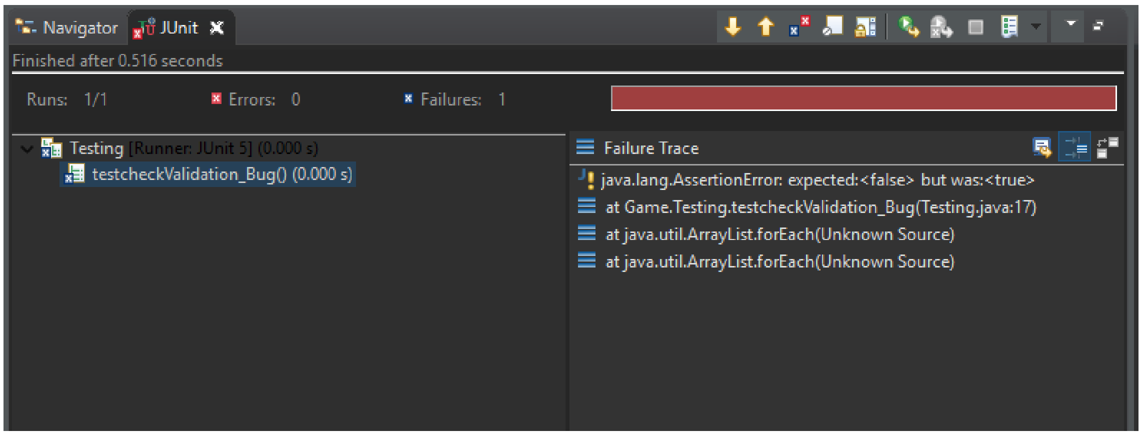
```
public boolean checkValidation(char n) {
    if(n == 'å')            // this is bug for testing
        n = 'a';
    switch(n){
    case 'A':
        chances=10;
        return true;
    case 'a':
        chances=10;
        return true;
    case 'B':
        chances=7;
        return true;
    case 'b':
        chances=7;
        return true;
    case 'C':
        chances =5;
        return true;
    case 'c':
        chances =5;
        return true;
    default:
        System.out.println("Invalid input!"+" Try again");
        return false;
    }
}
```

Junit test for S.code 3: Bug



```
    @Test
    public void testcheckValidation_Bug() { // it suppose to be false but it gives true as result
        assertEquals(false,difficulty.checkValidation('å'));

    }
```

Result of testing S.code 3: Bug



```
Finished after 0.516 seconds
Runs: 1/1    Errors: 0    Failures: 1

Testing [Runner: JUnit 5] (0.000 s)
    testcheckValidation_Bug() (0.000 s)

Failure Trace
java.lang.AssertionError: expected:<false> but was:<true>
at Game.Testing.testcheckValidation_Bug(Testing.java:17)
at java.util.ArrayList.forEach(Unknown Source)
at java.util.ArrayList.forEach(Unknown Source)
```

Reflection

The assignment has helped me to expand my learning in java. I have learnt new ways for developing the code and the concept of Inheritence was used in most of my classes. Though the assignment was quite challenging in the beginning, it was indeed a great way to learn the concept of testing. We have created a Test Plan and Manuel test cases. Moreover, junit testing was also implemented to the methods of various classes. I learnt that testing is an important tool in programming as the code needs to be flawless in order to successfully execute and hence meet the user requrements. Creating a bug and then making a junit case was a bit tricky but I managed to make it in the end. Overall, this assignment has enhanced my knowledge in programming and was quite interesting.