Test

Name: Mohammadali Rashidfarokhi

Email address: mr223jp@student.lnu.se

Link to my GitHub repo: https://gitlab.lnu.se/

Objective

The main purpose of this assignment is to test the provided code from the previous iteration which provided the player to choose random characters from a chosen word based on different levels.

What to test and How

I intended to test the methods that are suitable enough to be executed and some test to check its validity have been provided that run dynamic manual test-cases. At the beginning the TC1 will be evaluated. I also write automated unit tests for all player-methods in the class player test. However, as some methods cannot be used to write tests (e.g. the methods that will ask the player to provide some information and the ones that are not type void) in j unit all the methods in the class player will be used to write test cases for each one of them.

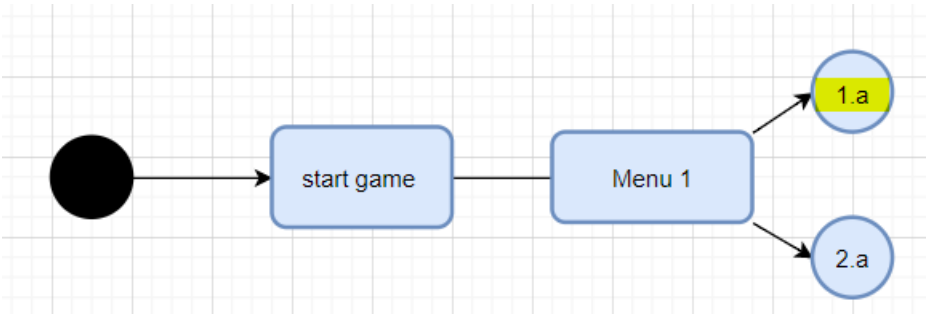
Time plan

Task	Estimated	Actual
Manual Tc	5 h	3 h
Unit tests	2 h	1:30 m
Running manual tests	1h	15 m
Code inspection	2 h	1 h
Test report	1 h	45 m

Manual Test-cases

Tc1.1 user choose to play the game

Scenario: the player chooses to continue (1.a) in Menu1.



The main scenario of this UC1.1 is tested where the user is faced with a menu (Menu 1) with two options and choose 1.a to continue.

Precondition: The player should not enter anything but 1.a

Test steps

• The player will run the game in terminal.

• User press 1.a in the keyboard.

• User press enter in the keyboard.

Expected

• Menu to choose the level appear.

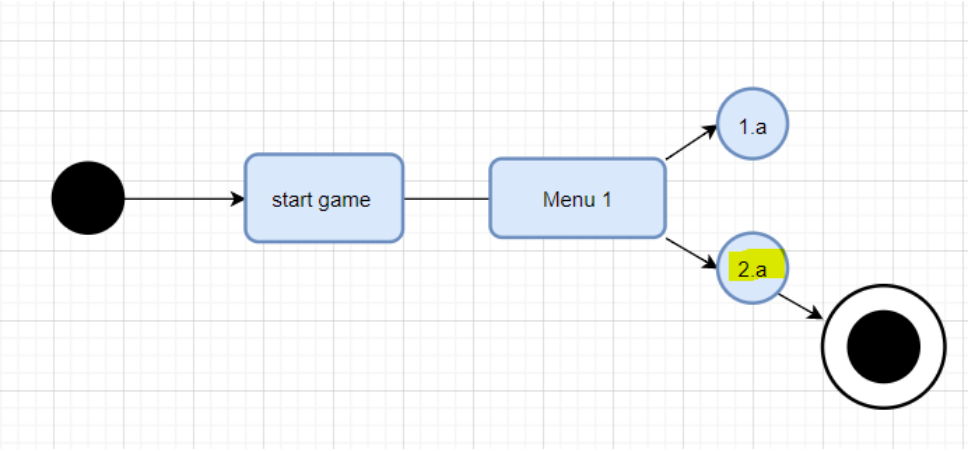
• The terminal shows the message “choose the level”.

• Press 1 for medium.

• Press 2 for hard

Tc1.2 User choose to leave the game.

Scenario: User will choose option 2.a (Exit game) in the menu1.



The main scenario of this UC1.1 is tested where the user is faced with a menu (Menu1) with two options and choose 2.a to Exit the game.

Precondition: The player should not enter anything but 2.a

Test steps

• The player will run the game in terminal.

• User press 2.a in the keyboard.

• User press enter in the keyboard.

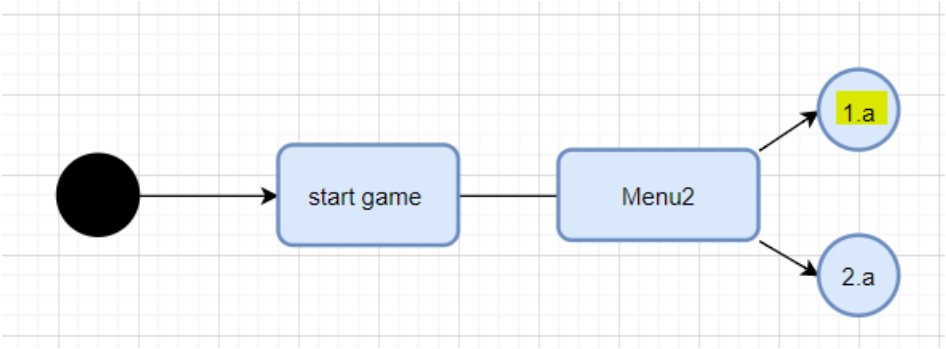
Expected:

• The system will show a goodbye message.

• The program ends.

TC2.1 user choose mode moderate

Scenario: The player choose option 1.a (moderate mode) in the menu2.



The main scenario of this UC2.1 is tested where the user is faced with a menu (Menu2) with two options and choose 1.a to set the game mode as moderate.

Precondition: The player should not enter anything but 1.a

Steps

• The player will run the program.

• User enters in keyboard to continue the game.

• User will be directed to choose level menu (menu 2).

• User will choose option 1.a in keyboard for moderate mode.

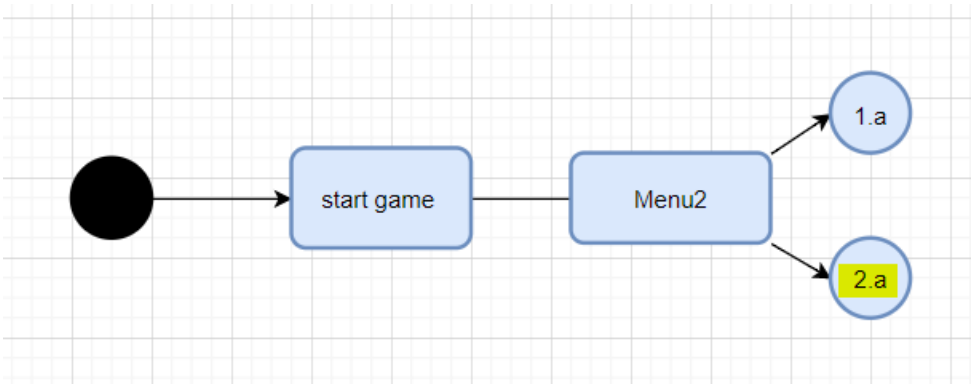
• User press enter in keyboard and set the game mode as moderate.

Expected

• The player will be given 8 attempts to find the missing letters.

TC2.2 user choose mode Hard

Scenario: The player choose option 2.a (Hard mode) in Menu2.



The main scenario of this UC2.2 is tested where the user is faced with a menu (Menu2) with two options and choose 2.a to set the game mode as Hard.

Precondition: The player should not enter anything but number 2.

Steps

- The player will run the program.

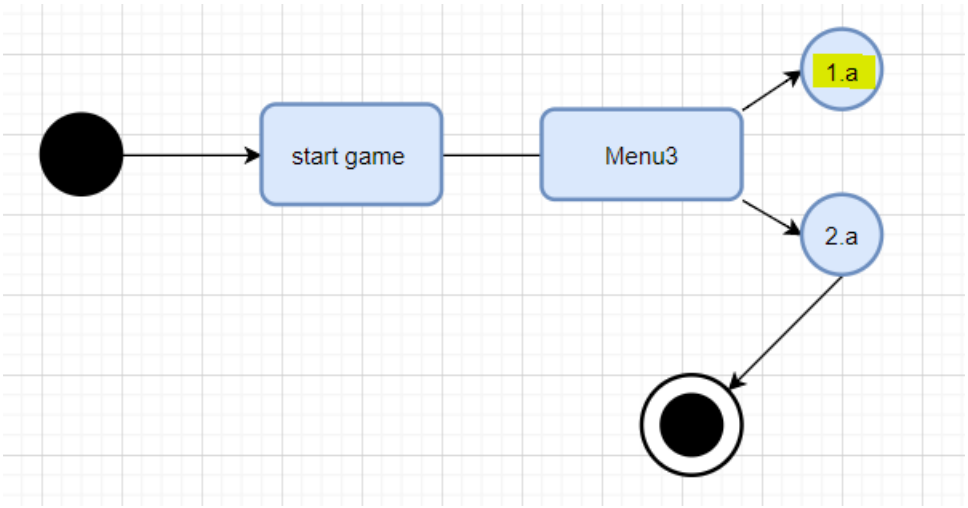
- User enters 1.a in keyboard to continue the game.
- User will be directed to choose level menu (menu 2).
- User will choose option 2.a in keyboard for hard mode.
- User press enter in keyboard and set the game mode as hard.

Expected

- The player will be given 6 attempts to find the missing letters.

TC3.1 user choose restart

Scenario: The player choose option 1.a (Restart game) in the Menu3.



The main scenario of this UC3.1 is tested where the user is faced with a menu3 with two options and choose 1.a to restart the game.

Precondition: The player should not enter anything but 1.a

Steps

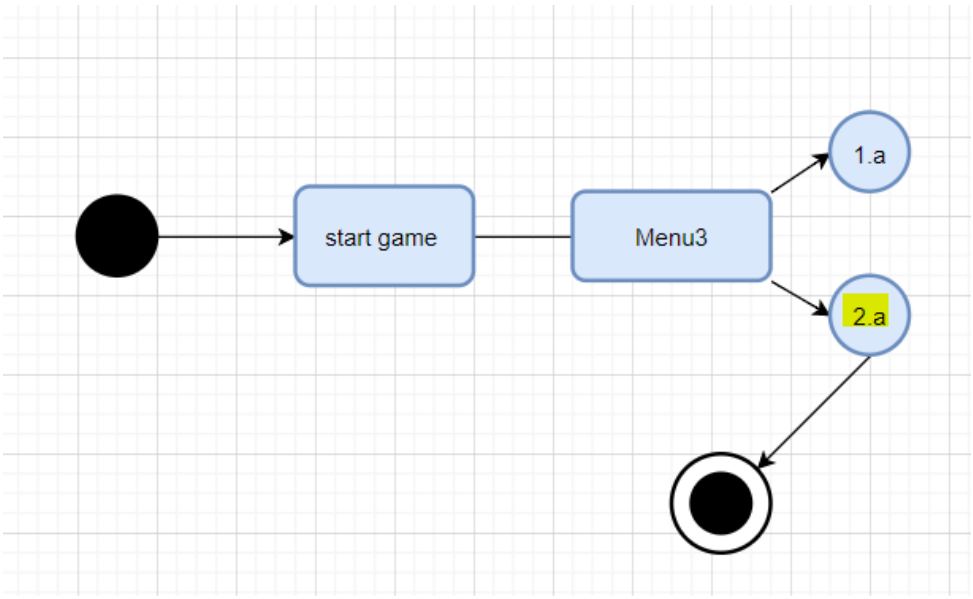
- The player will run the program.
- User enters 1.a in keyboard to continue the game.
- User will be directed to choose level menu (menu2).
- User will choose options 1.a or 2.a in keyboard for hard and moderate mode.
- User press enter in keyboard and set the game mode as hard or moderate.
- User will guess the missing letters.
- User will be directed to menu3.
- User will enter 1.a in keyboard to restart the game.

Expected

- The System will direct the user to the game menu stage.

TC3.2 user Exit the game

Scenario: The player choose option 2.a (Exit game) in the game menu3.



The main scenario of this UC3.2 is tested where the user is faced with a menu (menu3) with two options and choose 2.a to exit the game.

Precondition: The player should not enter anything but 2.a

Steps

- The player will run the program.
- User enters 1.a in keyboard to continue the game.
- User will be directed to choose level menu (menu2).

- User will choose options 1.a or 2.a in keyboard for hard and moderate mode.
- User press enter in keyboard and set the game mode as hard or moderate.
- User will guess the missing letters.
- User will be directed to the result menu (menu3).
- User will enter 2.a in keyboard to exit the game.

Expected

- The system will ask the user to enter some information.
- The system will provide user with few messages.
- The program ends.

Test reports

Test	UC1
TC1.1	1/OK
TC1.1	1/OK
Coverage & Success	2/OK

Test	UC2
TC2.1	1/OK
TC2.2	1/OK
Coverage & Success	2/OK

Test	UC3
TC3.1	1/OK
TC3.2	1/OK
Coverage & Success	2/OK

Test plan and unit test cases

Link to my code: https://gitlab.lnu.se/1dv600/student/mr223jp/assignment-3/-/blob/master/PlayerTest.java

```java
package Assingment3;

import org.junit.Test;
import org.junit.jupiter.api.BeforeEach;
import static org.junit.Assert.assertEquals;
import static org.junit.jupiter.api.Assertions.*;

public class PlayerTest {
    private Player testing = new Player();


    @Test
    public void setName() {
        String expected = "Ali", expected2 = "Marcos", expected3= "John",expected4 = "Emilia", expected5 = "Lana";
        String actual;

        testing.setName(expected);
        actual = testing.getName();
        assertEquals(expected, actual);

        testing.setName(expected2);
        actual = testing.getName();
        assertEquals(expected2, actual);

        testing.setName(expected3);
        actual = testing.getName();
        assertEquals(expected3, actual);

        testing.setName(expected4);
        actual = testing.getName();
        assertEquals(expected4, actual);

        testing.setName(expected5);
        actual = testing.getName();
```

PlayerTest.java | Player.java | GameMain.java | GameBrain.java

```java
        testing.setName(expected5);
        actual = testing.getName();
        assertEquals(expected5, actual);

    }

    @Test
    public void setAge() {
        int expected = 20,expected2 = 30, expected3= 40,expected4 = 50, expected5 = 60;

        testing.setAge(expected);
        int actual = testing.getAge();
        assertEquals(actual, expected);

        testing.setAge(expected2);
        actual = testing.getAge();
        assertEquals(actual, expected2);

        testing.setAge(expected3);
        actual = testing.getAge();
        assertEquals(actual, expected3);

        testing.setAge(expected4);
        actual = testing.getAge();
        assertEquals(actual, expected4);

        testing.setAge(expected5);
        actual = testing.getAge();
        assertEquals(actual, expected5);


    }

    @Test
```

Terminal | 0: Messages | Event Log

go) 266:115 CRLF UTF-8 master 4 spaces

```java
    @Test
    public void setDate()  {
        try {
            int expected = 2020;
            int expected2 = 11;
            int expected3 = 06;
            testing.setDate(expected, expected2, expected3);

            int expectedYear = 1999;
            int expectedMonth = 06;
            int expectedDay = 01;
            testing.setDate(expectedYear, expectedMonth, expectedDay);


            int expectedYear1 = 2020;
            int expectedMonth1 = 10;
            int expectedDay1 = 15;
            testing.setDate(expectedYear1, expectedMonth1, expectedDay1);


            int expectedYear2 = 1985;
            int expectedMonth2 = 05;
            int expectedDay2 = 29;
            testing.setDate(expectedYear2, expectedMonth2, expectedDay2);


            int expectedYear3 = 2019;
            int expectedMonth3 = 3;
            int expectedDay3= 25;
            testing.setDate(expectedYear3, expectedMonth3, expectedDay3);

        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
```

Terminal | 0: Messages | Event Log

go) 266:115 CRLF UTF-8 master 4 spaces

PlayerTest.java | Player.java | GameMain.java | GameBrain.java

```java
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    @Test
    public void setSurvey() {
        String expected = "I love the game", expected2 = "It was hard for me ", expected3= "It was amazing, loved it",
                expected4 = "Maybe its better to add some hints", expected5 = "Wow, it was absolutely a fun experience";

        testing.setSurvey(expected);
        String actual = testing.getSurvey();
        assertEquals(expected, actual);

        testing.setSurvey(expected2);
         actual = testing.getSurvey();
        assertEquals(expected2, actual);

        testing.setSurvey(expected3);
        actual = testing.getSurvey();
        assertEquals(expected3, actual);

        testing.setSurvey(expected4);
        actual = testing.getSurvey();
        assertEquals(expected4, actual);

        testing.setSurvey(expected5);
        actual = testing.getSurvey();
        assertEquals(expected5, actual);
    }

    @Test
    public void getName() {
```

Terminal | 0: Messages | Event Log

go) 266:115 CRLF UTF-8 master 4 spaces

```java
    @Test
    public void getName() {
        String expected = "Johnny", expected2 = "Victor", expected3= "Bryan",expected4 = "John snow", expected5 = "Mother of dragons";

        testing.setName(expected);
        String actual = testing.getName();
        assertEquals(expected, actual);

        testing.setName(expected2);
        actual = testing.getName();
        assertEquals(expected2, actual);

        testing.setName(expected3);
        actual = testing.getName();
        assertEquals(expected3, actual);

        testing.setName(expected4);
        actual = testing.getName();
        assertEquals(expected4, actual);

        testing.setName(expected5);
        actual = testing.getName();
        assertEquals(expected5, actual);
    }

    @Test
    public void getAge() {
        int expected = 12,expected2 = 35, expected3= 25,expected4 = 65, expected5 =90;
```

Terminal | 0: Messages | Event Log

go) 266:115 CRLF UTF-8 master 4 spaces

```java
    }

    @Test
    public void getAge() {
        int expected = 12,expected2 = 35, expected3= 25,expected4 = 65, expected5 =90;

        testing.setAge(expected);
        int actual = testing.getAge();
        assertEquals(expected, actual);

        testing.setAge(expected2);
         actual = testing.getAge();
        assertEquals(expected2, actual);

        testing.setAge(expected3);
        actual = testing.getAge();
        assertEquals(expected3, actual);

        testing.setAge(expected4);
        actual = testing.getAge();
        assertEquals(expected4, actual);

        testing.setAge(expected5);
        actual = testing.getAge();
        assertEquals(expected5, actual);
    }
```

```java
    }

    @Test
    public void getSurvey() {
        String expected = "was nice", expected2 = "SoSO", expected3= "Such an amazing game",
                expected4 = "Didn't like the game", expected5 = "Boring";


        testing.setSurvey(expected);
        String actual = testing.getSurvey();
        assertEquals(actual, expected);

        testing.setSurvey(expected2);
        actual = testing.getSurvey();
        assertEquals(actual, expected2);

        testing.setSurvey(expected3);
        actual = testing.getSurvey();
        assertEquals(actual, expected3);
    }
```

```java
    @Test
    public void getDate() throws Exception {
        try {

            int year = 2020, month = 05, day = 06;
            testing.setDate(year, month, day);
            String actual = testing.getDate();
            assertEquals( expected: day + "/" + month + "/" + year, actual);

            int year1 = 2019, month1 = 10, day1 = 13;
            testing.setDate(year1, month1, day1);
             actual = testing.getDate();
            assertEquals( expected: day1 + "/" + month1 + "/" + year1, actual);

            int year2 = 2018, month2 = 12, day2 = 28;
            testing.setDate(year2, month2, day2);
            actual = testing.getDate();
            assertEquals( expected: day2 + "/" + month2 + "/" + year2, actual);

            int year3 = 1995, month3 = 03, day3 = 01;
            testing.setDate(year3, month3, day3);
            actual = testing.getDate();
            assertEquals( expected: day3 + "/" + month3 + "/" + year3, actual);

            int year4 = 1999, month4 = 07, day4 = 30;
            testing.setDate(year4, month4, day4);
            actual = testing.getDate();
            assertEquals( expected: day4 + "/" + month4 + "/" + year4, actual);

        }catch (Exception e){
            System.out.println(e.getMessage());
        }
    }
```

```java
    @Test
    public void setValidDate() {
        int expected = 2020;
        int expected2 = 11;
        int expected3 = 06;
        try {
            testing.setDate(expected, expected2, expected3);
        }catch (Exception e) {
            System.err.println(e.getMessage());
        }
        assertEquals( expected: expected3 + "/" + expected2 + "/" + expected, testing.getDate());

        expected = 1378;
        expected2 = 01;
        expected3 = 4;
        try {
            testing.setDate(expected, expected2, expected3);
        }catch (Exception e) {
            System.err.println(e.getMessage());
        }
        assertEquals( expected: expected3 + "/" + expected2 + "/" + expected, testing.getDate());
    }
    @Test
    public void setInvalidDate() {
        assertThrows(Exception.class,() -> {
            testing.setDate( year: 2020,  month: 13,  day: 06);
        });

        assertThrows(Exception.class,() -> {
            testing.setDate( year: 2020,  month: 01,  day: 40);
        });
    }
    @Test
```

```
PlayerTest.java  Player.java  GameMain.java  GameBrain.java
            testing.setDate(expected, expected2, expected3);
        }catch (Exception e) {
            System.err.println(e.getMessage());
        }
        assertEquals( expected: expected3 + "/" + expected2 + "/" + expected, testing.getDate());
    }
    @Test
    public void setInvalidDate() {
        assertThrows(Exception.class,() -> {
            testing.setDate( year: 2020,  month: 13,  day: 06);
        });

        assertThrows(Exception.class,() -> {
            testing.setDate( year: 2020,  month: 01,  day: 40);
        });
    }
    @Test
    public void checkAli() {
        assertTrue(testing.isAli( s: "Ali"));
        // This test will pass if the 3 lines of codes that I put in comment in player class in the lines 56,57,58 are commented out.
    }

    @Test
    public void checkJack() {
        assertFalse(testing.isAli( s: "Jack"));

    }
}
```

Automated unit test code screenshots

The below pictures illustrate two test codes like each other but one of them will pass and the other will fail when executing at the same time.

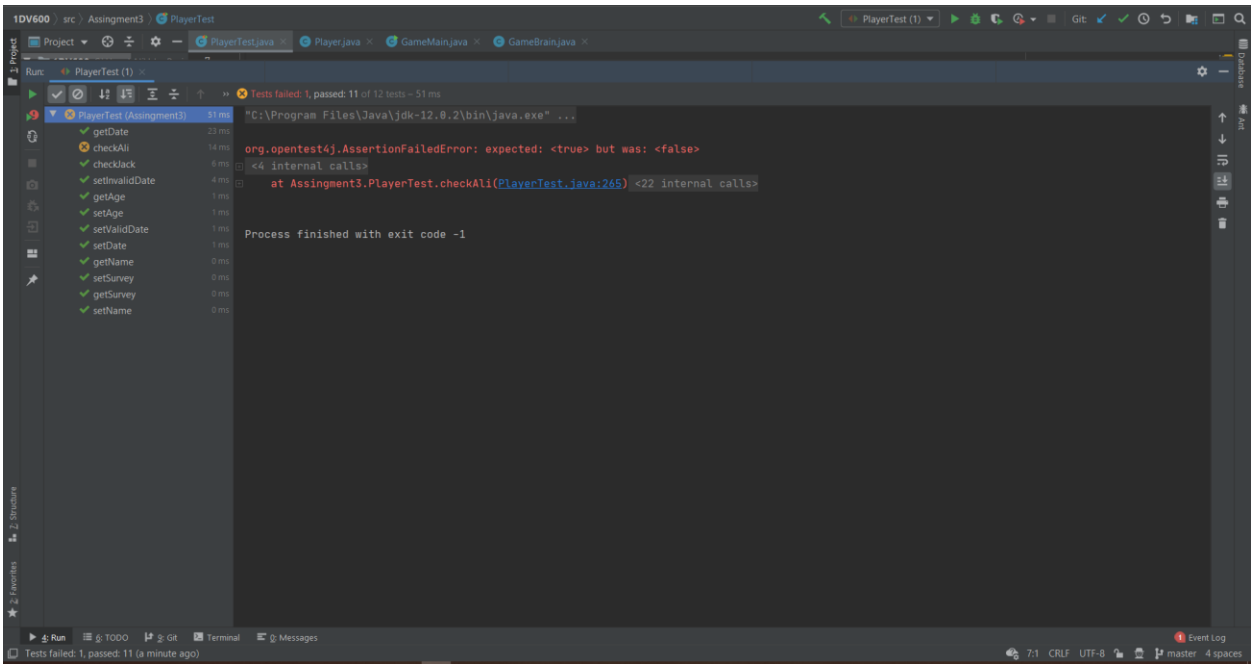
The test method called checkAli(), this Test can pass if the codes in lines 56,57,58 in class player are commented out.

```
    }
    @Test
    public void checkAli() {
        assertTrue(testing.isAli( s: "Ali"));
        // This test will pass if the 3 lines of codes that I put in comment in player class in the lines 56,57,58 are commented out.
    }

    @Test
    public void checkJack() {
        assertFalse(testing.isAli( s: "Jack"));

    }
}
```

As I have mentioned before the Test checkAli() will fail and the Test checkJack() will pass.



Comments

All the tests passed, but more test for data user case can be written to test if it works or not. However, two methods called checkAli() and checkJack() are the prominent examples that will demonstrate one test will fail on purpose and the other will pass (checkJack).

Reflection

In this iteration, 3 use cases have been used to write a manual test case, the results were promising since the tests passed. However, more use cases can be written to make sure if they can pass or fail. An example would be writing a use case along with 2 test cases for the data stage in my code.

Regarding the automated test case in my program, there are 10 tests in total which all will pass except one of them (fail on purpose).

A noticeable point about the test that fails (checkAli) is that there is another test called checkJack() which is similar to the checkAli() test, but one will fail and the other will pass.

I have spent more than four hours to come up with the test that fails on purpose. In the beginning, my idea about this test was to make some changes to the date method in the player. But I came with the idea of creating two methods that fail and pass at the same time. Consequently, I spent more time on creating these two methods.

Thanks to this assignment I have improved my skills at writing tests in junit and from this assignment, a developer can make sure either the methods are going to work or not. But it would be better there was more time to spend on this project. Therefore, more precise tests would have been written.