Test Plan

Objective

The objective is checking the results of the test run for errors or information about the program's non-functional attributes by executing the program that was implemented.

What to test and how.

We intend to test UC1("Start Game") and UC2("Play Game") by writing and running dynamic manual test-cases.We also write automated unit tests for isContains( ), selectionSort( ) and isExist( ) methods.Finally we examine the code by static inspection to see how we can fix the errors.

Time plan

Task	Estimated	Actual
Planning the Assignment	30m	1h
Manual Test Cases	2h	3h
Running manual tests	20m	5m
Unit Test	2h	3h
Test Report	1h	2h

Manual Test Cases

TC1.1 Register and Start game

Use case: UC1("Start Game")

Scenario: The main scenario of UC1 is tested where the user registers by writing a new username and password correctly and logins the game(perfect scenario)

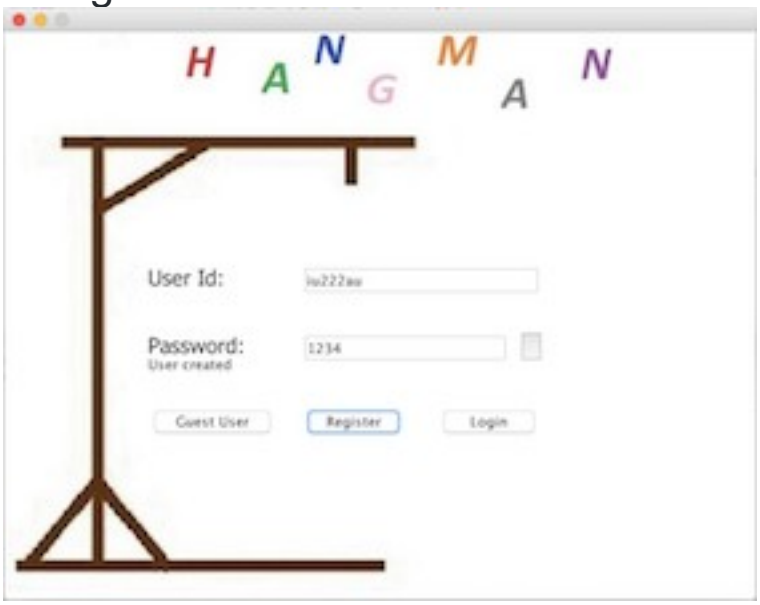
Precondition:There must NOT be a username which is like "iu222au" in the DataSet.txt file.

Test steps

- Start the app
- Enter "iu222au" as username and "1234" as password.
- Press "Register" button
- Press "Login" button

Expected

- The system shows "User created" and user got ability to click the "Login" button.



TC1.2 Can not Register because of Same Username

Use case: UC1("Start Game")

Scenario: User enters a username that is already exist and can not register to game.

Precondition:There must be a username "iu222au" in the DataSet.txt file.

Test steps

- Start the app
- Enter "iu222au" as username and "5678" as password.
- Press "Register" button
- Press "Login" button

Expected

- The system shows a message: "This user already exist!" and waits for new username and password

TC1.3 Can not Login because of Wrong Password

Use case: UC1("Start Game")

Scenario: User enters a username and password but can not login game because of the wrong password.

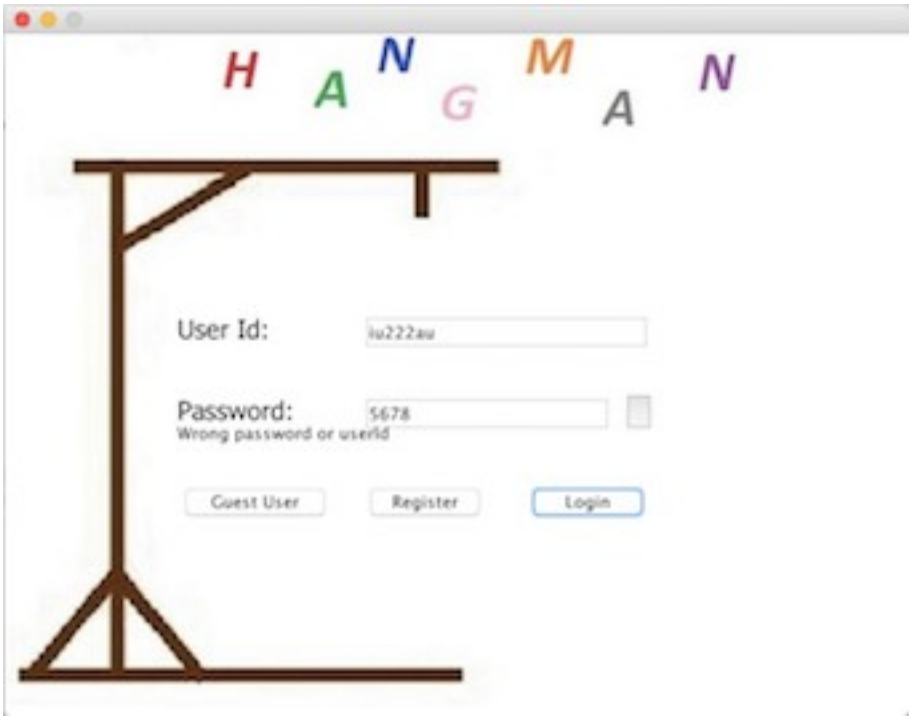
Precondition:There must be a username "iu222au" with password "1234".

Test steps

- Start the app
- Enter "iu222au" as username and "5678" as password.
- Press "Login" button

Expected

- The system shows a message: "Wrong password or userId" and waits for correct username and password



Note: The word is set as "classic" for the test case

TC2.1 Play game and Win

Use case: UC2("Play Game")

Scenario: The main scenario of UC2 is tested where the user plays the game, guesses every letters correctly and wins the game.(perfect scenario)

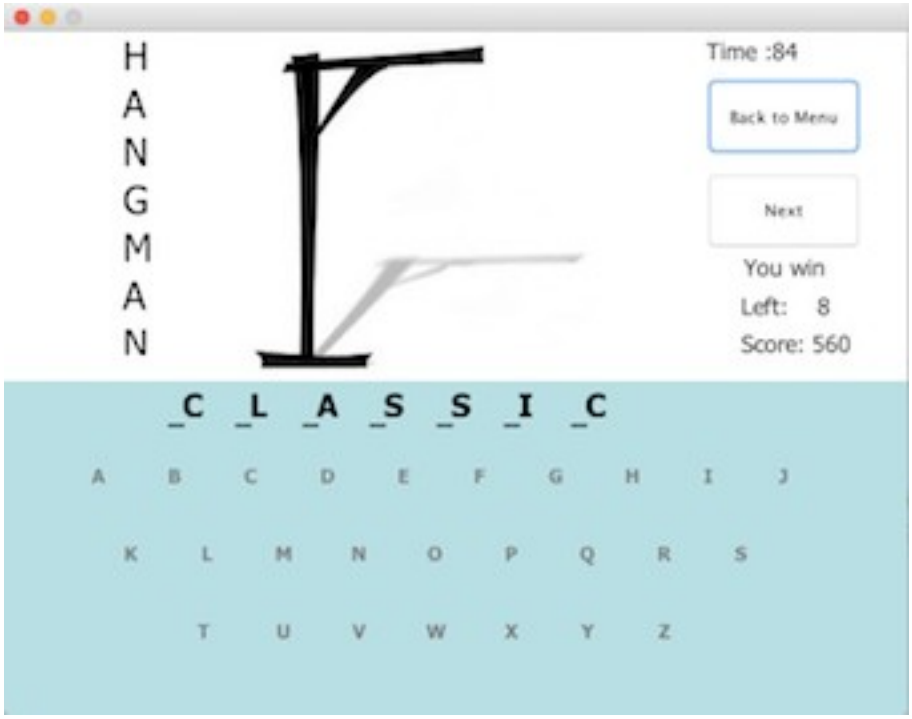
Precondition:UC1

Test steps

- Press the button "Play Game"
- Press the button 'c'
- Press the button 'l'
- Press the button 'a'
- Press the button 's'
- Press the button 'i'
-

Expected

- The system shows "You Win" and displays a new button "Next" and the time continues to flow for next word.



TC2.2 Play game and Loose with no guess chance

Use case: UC2("Play Game")

Scenario: The user plays the game, guesses letters wrongly and there is no chance to guess a word and lost game.

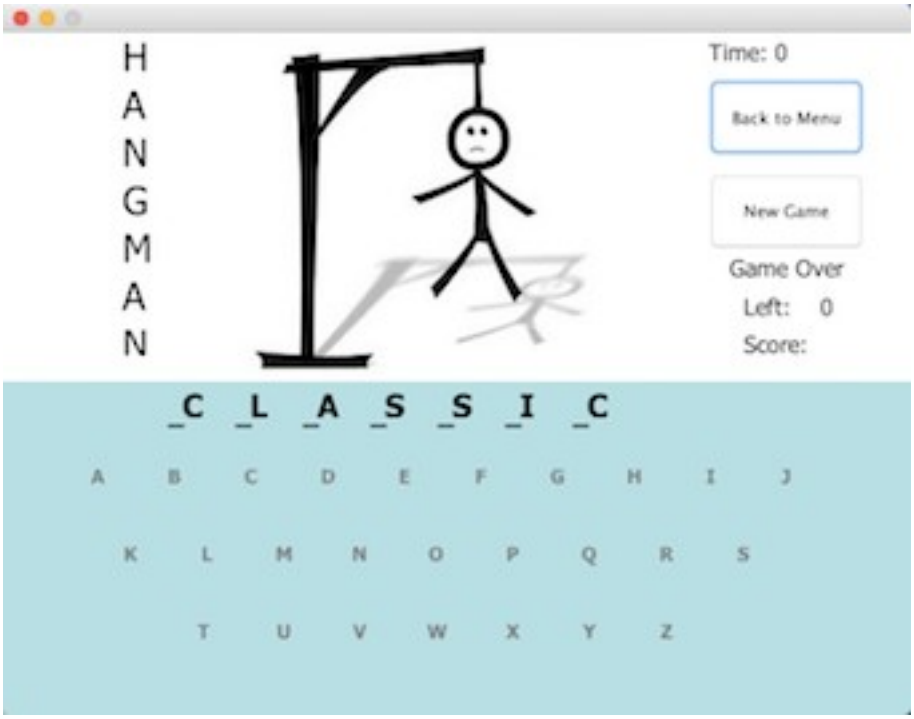
Precondition:UC1

Test steps

- Press the button "Play Game"
- Press the button 'b'
- Press the button 'd'
- Press the button 'e'
- Press the button 'f'
- Press the button 'g'
- Press the button 'h'
- Press the button 'j'
- Press the button 'k'

Expected

- The system shows "Game Over", "time : 0" ,"left : 0" and "score: ".
- Displays a new button "New Game" and the true word.
- The system shows the hanged man.



Test Report

Test	UC1	UC2
TC1.1	1/OK	0
TC1.2	1/OK	0
TC1.3	1/OK	0
TC2.1	0	1/OK
TC.2.2	0	1/OK
COVERAGE & SUCCESS	3/OK	2/OK

Unit Test

```
14  public class Testing {
15
16      Hangman hm;
17      Menu menu;
18      Login login;
19
20      @Before
21      public void setUp() throws IOException{
22          hm = new Hangman();
23          menu = new Menu();
24          login = new Login();
25      }
26
27
28
29
30      @Test
31      public void testIsExist() {
32
33          boolean expected = true;
34          boolean actual = login.isExist("iu222au");
35
36          assertEquals(expected,actual);
37
38      }
39
40
41      @Test
42      public void testForIsContains() throws IOException {
43
44          boolean expected=true;
45          boolean actual = hm.isContains('e','E',"Egemen");
46
47          assertEquals(expected,actual);
48      }
49
50      @Test
51      public void testForIsContainsNumber() throws IOException {
52
53          boolean expected=false;
54          boolean actual = hm.isContains('1','3',"Egemen");
55
56          assertEquals(expected,actual);
57      }
```

```java
    @Test
    public void testSortForEmptyAndSingleton() {

        int[] a1 = {};
        String[] b1 = {};
        int[] arr1 = new int[0];            //Empty array
        menu.selectionSort(a1,b1);
        assertEquals(0,a1.length);

        int[] a2 = {160};                   //Singleton array
        String[] b2 = {"iu222au"};
        menu.selectionSort(a2, b2);
        assertEquals(1,a2.length);
        assertEquals(160,a2[0]);
        assertEquals("iu222au",b2[0]);

    }

    @Test
    public void testSortFortEqualAndRandom(){

        int[] a3 = {160,160,160};                   //All elements are equal
        String[] b3 = {"iu222au","ipek","berk"};
        menu.selectionSort(a3, b3);
        assertEquals(3,a3.length);
        for(int i : a3)
            assertEquals(160,i);

        int[] a4 = random(10,10);               //Random array of size 10
        String[] b4 = {"a","b","c","d","e","f","g","h","i","j"};
        menu.selectionSort(a4, b4);
        for(int i=0; i<a4.length-1; i++)
            assertTrue(a4[i]>=a4[i+1]);


    `
    /*
     * Generates random integer array of length size with elements
     * in the range [0,max].
     */

    private int[] random(int size,int max) {
        /* Setup random generator */
        Random rand = new Random();

        /* Add random numbers in range [1,max] */
        int[] arr = new int[size];
        for (int i=0;i<size;i++) {
            //int n = rand.nextInt() % max;
            int n = rand.nextInt(max);
            arr[i] = n;
        }
        return arr;
    }


}
```

Package Explorer JUnit

Finished after 3,401 seconds

Runs: 5/5 Errors: 0 Failures: 1

- iu222au.Testing [Runner: JUnit 4] (3,354 s)
  - testSortForEmptyAndSingleton (3,013 s)
  - testSortFortEqualAndRandom (0,071 s)
  - testForIsContainsNumber (0,110 s)
  - testForIsContains (0,074 s)
  - testIsExist (0,086 s)

Failure Trace

```
java.lang.AssertionError: expected:<true> but was:<false>
at iu222au.Testing.testIsExist(Testing.java:38)
```

Coverage

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
1DV600	62,7 %	2.558	1.524	4.082
src	62,7 %	2.558	1.524	4.082
iu222au	62,7 %	2.558	1.524	4.082
Hangman.java	65,1 %	1.591	853	2.444
Hangman	80,0 %	1.305	327	1.632
pressedButton(JButton, char, ch...	0,0 %	0	228	228
Hangman()	94,1 %	1.080	68	1.148
addScore()	0,0 %	0	31	31
buttonEnable(boolean)	100,0 %	105	0	105
createrButton(JButton, int, int)	100,0 %	25	0	25
createrLabels(JLabel[], String, b...	100,0 %	45	0	45
isContains(char, char, String)	100,0 %	22	0	22
selectedWord()	100,0 %	5	0	5
startTimer(int)	100,0 %	23	0	23
Menu.java	55,6 %	404	323	727
Menu	98,7 %	377	5	382
Menu()	99,1 %	318	3	321
getLevel()	0,0 %	0	2	2
selectionSort(int[], String[])	100,0 %	59	0	59
Login.java	54,5 %	292	244	536
Login	96,1 %	268	11	279
isExist(String)	63,3 %	19	11	30
Login()	100,0 %	249	0	249
Main.java	0,0 %	0	97	97
Testing.java	97,5 %	271	7	278
Testing	97,5 %	271	7	278

Reflection

It was little bit difficult to planning the test. Because it was my first time writing a manual test and it is hard to predict the time of something that you do not know anything about.
For manual test, it took a bit time at the beginning. But after i learnt about it, it was easy to implement and fun. the Greeter example was so helpful for that. And also was fun to could take screenshots from the results.
Because of the fact that i did not know how to test the GUI, learning the automated test and finding separate methods were also difficult for me. I tried details and tested them.
After all, this part of the assignment was so instructive and even if i spent lots of time to learn something, i like what i learnt and i think they are so useful to realize the details.