Marcus Gärdskog Hill

Mg223hh

Iteration 3, Testing

mg223hh@student.lnu.se

https://github.com/MarcusGHill/mg223hh1dv600

Test Plan

Project hangman

Objective

The objective is to test part of the code written so far for the project hangman application according to the instructions at https://github.com/dntoll/1dv600/tree/master/A3.

What to test and how

All tests in this report is dynamic and done by the author, first I will manually test the "Choose wikipedia site to fetch words from" refered to as “wikipedia” in this report and the "quit game" use case. The wikipedia use case has a lot of moving parts and is best tested manually so that I can make sure that I have internet connection and if it takes a lot of time I can manually check what might be going on. Also I am scraping wikipedia and I don't want to automate it and accidentally make to many calls and get my IP blocked. The quit game use case will also be manually tested because it is a easy enough test to do manually, there is not to many possible outcomes and It won't take to much time.

Then I will write two Junit tests each for two methods. The first method is addHighScore() in the HighScore class to check that the highscores get added and sorted properly so that the highscore can be displayed in the correct way. This method works well for automated testing as it got simple inputs and is easy to check if it performed correctly. The second method that I will create Junit tests for is initializeHangman() in the gameLogic class. The tests will make sure that the hangman get initialized in the correct form and look exactly the same every time. This method is also well suited for automated testing since I might miss small changes in the hangman "image" by just looking at it but the automated test will find any deviation. Lastly one automated Junit test called checkNickNameTest() that will detect a failure in a incomplete method for checking if a nickname is saved.

Time plan

Task	Estimated time	Actual time
Manual test case	2 hours	8 hours
Unit test	3 hours	9 hours
Running manual test	1 hours	1 hours
Create test report	5 hours	15 hours

Manual Test-Cases

TC1 Choose wikipedia site to fetch words from

Use case: **Choose wikipedia site to fetch words from**

What's tested: The applications ability to fetch words for the game from a English wikipedia site.

Scenario: Successfully fetch words from chosen wikipedia

Precondition: Active internet connection and application started

Test steps

1. Run the app in IDE

System shows:

"--welcome, to start game type start and press Enter

--if you want to choose what wikipedia site to fetch your words from type the address

and press enter to start the game. Type quit at any time terminate the application

Example https://en.wikipedia.org/wiki/Stockholm"

2. Enter the address " https://en.wikipedia.org/wiki/Stockholm" and press enter

Game starts and shows:

amount of wrong guesses = 0

your wrong guesses =

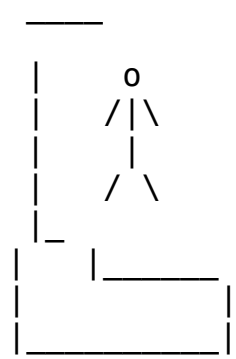
.....

Take a guess!

3.Push q and hit enter

Repeat step 3 changing q for w,t,p,s,d,g,h,k,l,z,x,c,v,b,n,m,until game finish

Game shows:



```
GAME OVER!, hit Enter to restart application, the answer was
continental
```

4. Copy the word after "the answer was " and go to https://en.wikipedia.org/wiki/Stockholm

5. crl + F Search for the word to see that the word got pulled from the page.

Expected

The crl + F search should find the word on the page

Game asks user to hit enter to restart application

Test successfull	Test unsuccessfull
OK	

Note:
Test was successfull every time it was conducted

TC1.2 invalid adress prompt user for new adress

Use case: **Choose wikipedia site to fetch words from**

What's tested: The applications ability recognize an incomplete address and tell the user to try again.

Scenario: unsuccessful to fetch words from the address given.

Precondition: user is at the main menu.

Test steps

1.Run the app in IDE

•System shows:
"`--welcome, to start game type start and press Enter`

`--if you want to choose what wikipedia site to fetch your words from type the adress and press enter to start the game. Type quit at any time terminate the application Example https://en.wikipedia.org/wiki/Stockholm`"

2.Enter the adress “ `ht://en.wikipedia.org/wiki/nothingheretosee`” and press enter

•**Expected**

•The system should show the text "`the URL does not look right, try again and follow the format https://en.wikipedia.org/wiki/Stockholm`"

•System shows main menu again and waits for input.

Test successfull	Test unsuccessfull
OK	

Note:
Test was successfull every time it was conducted

TC2 Quit game

Use case: Quit game

What's tested: The users ability to terminate the application from the main menu.

Scenario: terminate the application from main menu

Precondition: user is at the main menu.

Test steps

1. Run the app in IDE

•System shows:
"--welcome, to start game type start and press Enter

--if you want to choose what wikipedia site to fetch your words from type the adress
and press enter to start the game. Type quit at any time terminate the application
Example https://en.wikipedia.org/wiki/Stockholm"

2.Enter “quit” and hit enter

3.systems shows “Are you sure you want to terminate? Yes/no”

4.Enter “yes” and hit enter

Expected

•The system should terminate the application.

Test successfull	Test unsuccessfull
OK	

Note: This should work at any time during the game and does not have to be done at main menu.

TC2.1 Quit game, change mind.

Use case: Quit game.

Whats tested: Terminate the application from the main menu but change mind.

Scenario: starting termination of the application from main menu when writing “quit” and hitting enter then typing “no” and hitting enter to return to main menu.

Precondition: user is at the main menu.

Test steps

1.Run the app in IDE

•System shows:
"--welcome, to start game type start and press Enter

--if you want to choose what wikipedia site to fetch your words from type the adress
and press enter to start the game. Type quit at any time terminate the application
Example https://en.wikipedia.org/wiki/Stockholm"

2.Enter “quit” and hit enter

3.systems shows “Are you sure you want to terminate? Yes/no”

4.Enter “no” and hit enter
Expected

•The system should return to main menu and await input.

Test successfull	Test unsuccessfull
OK	

Note:

1.Anything but “yes” will return user to main menu.

2.This will work at any time during the game and does not have to be done at main menu.

Test Report

Test traceability and success matrix

Test	UC Wiki	UC Quit
TC1.1	OK	

TC1.2	OK	
TC2.1		OK
TC2.2		OK
COVERAGE & SUCCESS	2/OK	2/OK

Automated unit test coverage and success

Test	Method tested	Coverage & success
InitializeHangmanTest	initializeHangman()	100/OK
testAddHighScore	HighScore.addHighScore()	100/OK
checkNickNamesTest	checkNickNames	100/FAIL

Comment

All tests that tested complete methods pass, next iteration I need to focus on extending use cases and if needed refactor the code.

Automated Junit test cases

Test case 1:

Whats tested: The applications ability to create the hangman in the right porportions
Precondition: application started

Coverage, initializeHangmanTest():

```
    // Populates the hangman Array list with the hangman
public void initializeHangMan () {
        hangman.add("   ____");
        hangman.add( "  |    o  ");
        hangman.add( "  |   /|\\   ");
        hangman.add( "  |    |  ");
        hangman.add( "  |   / \\");
        hangman.add( " _|_ ");
        hangman.add( "|   |_____ ");
        hangman.add( "|         |  ");
        hangman.add( "|_________| ");
}
```

Test code, initializeHangmanTest():

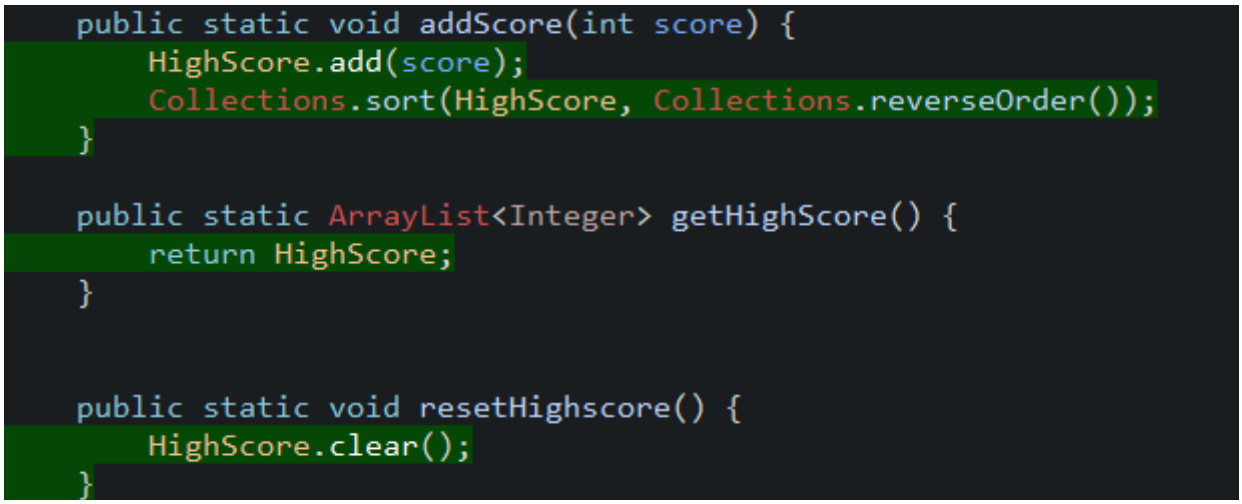
```java
    // Test to make sure the hangman arraylist gets properly constructed
    @Test
    void testInitializeHangMan() {
        HangmanLogic hl = new HangmanLogic();
        hl.initializeHangMan();
        String Expected0fromTop = "    ____";
        String Expected1FromTop = "  |    o  ";
        String Expected2FromTop = "  |   /|\\   ";
        String Expected3FromTop = "  |    |  ";
        String Expected4FromTop = "  |   / \\";
        String Expected5FromTop = " _|_ ";
        String Expected6FromTop = "|   |_____ ";
        String Expected7FromTop = "|         |  ";
        String Expected8FromTop = "|_________| ";
        assertEquals(hl.getHangman(0), Expected0fromTop);
        assertEquals(hl.getHangman(1), Expected1FromTop);
        assertEquals(hl.getHangman(2), Expected2FromTop);
        assertEquals(hl.getHangman(3), Expected3FromTop);
        assertEquals(hl.getHangman(4), Expected4FromTop);
        assertEquals(hl.getHangman(5), Expected5FromTop);
        assertEquals(hl.getHangman(6), Expected6FromTop);
        assertEquals(hl.getHangman(7), Expected7FromTop);
        assertEquals(hl.getHangman(8), Expected8FromTop);
    }


// Test to make sure that the index out of bounds exception gets thrown if hangman
//  is not initialized or initialized but called with Index 9 (its only 0-8)

    @Test
    public void testIndexOutOfBoundsException() {
        HangmanLogic hl = new HangmanLogic();
        Assertions.assertThrows(IndexOutOfBoundsException.class, () -> hl.getHangman(0));
        hl.initializeHangMan();
        Assertions.assertThrows(IndexOutOfBoundsException.class, () -> hl.getHangman(9));
    }
```

Test case 2:

Whats tested: To add to highscore and keep highscore sorted

Precondition: application started

coverage, testAddHighScore():



Test code part 1, testAddHighScore():

```java
    // Testing that highScore gets added
    @Test
    void testAddHighScore() {
        ArrayList<Integer> HScore = new ArrayList<Integer> ();
        int one = 1;
        int two = 2;
        int four = 4;
        int twenty = 20;
        int nintey = 90;
        int ten = 10;
        HighScore.addScore(four);
        HighScore.addScore(twenty);
        HighScore.addScore(nintey);
        HighScore.addScore(two);
        HighScore.addScore(one);
        HScore = HighScore.getHighScore();
        int accualSize = HScore.size();
        int expectedSize = HScore.size();
        // Test that all integers are there
        assertEquals(accualSize, expectedSize);
        HighScore.addScore(ten);
         accualSize = HScore.size();
        expectedSize = accualSize;
        assertEquals(expectedSize, accualSize);

        // Test max and min
        int maxInHScore = Collections.max(HScore);
        int minInHScore = Collections.min(HScore);
        assertEquals(maxInHScore, nintey);
        assertEquals(minInHScore, one);
    }
```

Test code part 2, testAddHighScore():

```java
// Testing that highscore gets sorted
@Test
void testSortHighScore() {
    HighScore.resetHighscore();
    ArrayList<Integer> HScore2 = new ArrayList<Integer> ();
    HScore2 = HighScore.getHighScore();
    int one = 1;
    int six = 6;
    int seven = 7;
    int eight = 8;
    int nine = 9;
    int ten = 10;
    HighScore.addScore(seven);
    HighScore.addScore(eight);
    HighScore.addScore(nine);
    HighScore.addScore(one);
    HighScore.addScore(six);
    HighScore.addScore(ten);
    int firstPlace = HScore2.get(0);
    int lastPlace = HScore2.get(5);
    assertEquals(firstPlace, ten);
    assertEquals(lastPlace, one);

    // Check that its actually sorted
    for(int i = 0; i < HScore2.size() - 1; i++) {
        assertTrue(HScore2.get(i) > HScore2.get(i+1));
    }
```

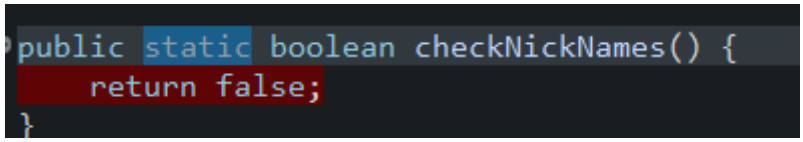
Test case 3:

Whats tested: the functionality to see if a Nickname is present.

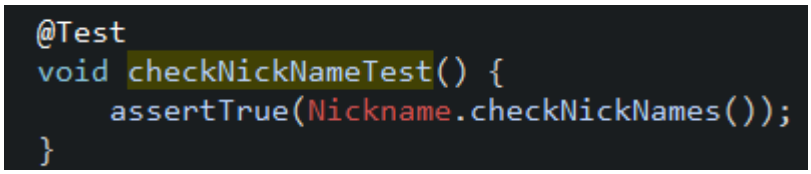
Precondition: application started

checkNickNamesTest:

coverage:



Test code:



Reflection:

Before this iteration the word testing sounded really boring to me and I thought it was something I would not like, how ever this iteration has been eye opening for me. My code in this project is not very great. It's not written to be tested since I have never really tested before I did not think about it. I have gained a lot of understanding of what I need to do differently when I code and I'm looking forward to writing code with testing in mind since it will be so much better and I will have alot more confidence in the code I produce.