Software Quality – Assignment 1

https://github.com/JadEletry/Hangman

Reason of Choice

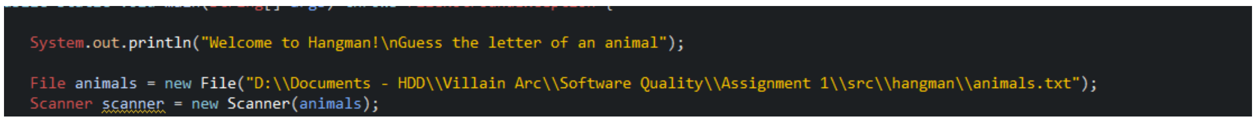
When planning my approach to creating hangman I decided to follow along the incremental software process as it was simple and straightforward like the program I was planning to create. Since the game I had in mind wasn't particularly too complex to create, I decided it was more effective to follow the incremental process since it reduces the cost of accommodating changes to my software and therefore requirements can be manipulated quite easily. I took a more agile approach when following this method as it helped me maintain simplicity and focus on the software being developed as well as actively eliminating complexity from the system.

The incremental model modularizes my initial requirements for my system into multiple standalone features which were easier to implement since I only needed to focus on the design & development of a single component at a time. After finishing the design & development phase of each component of the software, I began the testing of all methods in the program. Modularization helped ease the testing phase as I was able to successfully pass each method one by one making them ready for implementation.

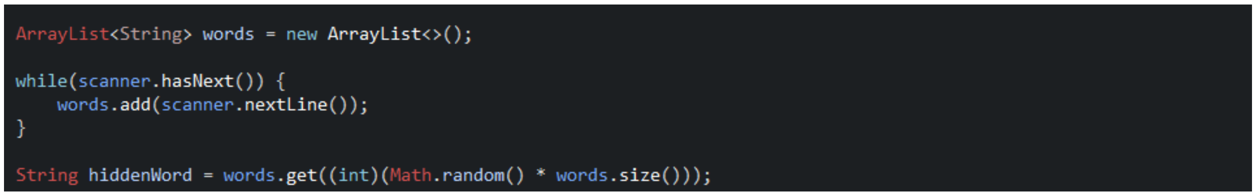
Description of Functionalities

Hangman.java is exactly what the name refers to - a simple game of hangman. The software was created in Eclipse using java and was tested using the Junit testing framework which is built in with the Eclipse workspace. This form of hangman is categorized with animals, where the user is to guess each letter of a randomized animal in order to win. The user has a total of 6 guesses where if they fail to guess the animal in the given number of guesses or "lives" then he/she will die, hence the word hangman. It's a bit brutal however that's the game itself.

The system uses a txt file filled with random animals listed alphabetically. Basically, this is how the program generates a random animal – by utilizing the Scanner function using the nextLine() input type and reading a random animal through the Random library.

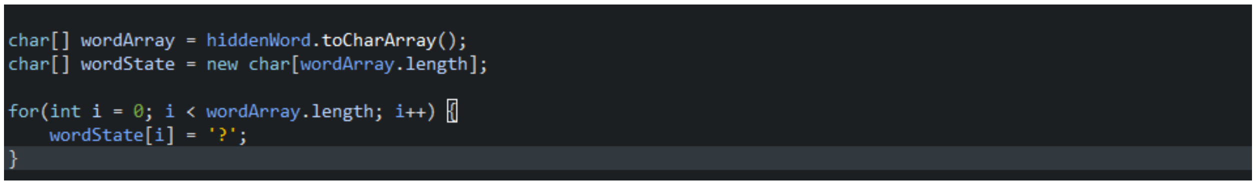


```
  System.out.println("Welcome to Hangman!\nGuess the letter of an animal");

  File animals = new File("D:\\Documents - HDD\\Villain Arc\\Software Quality\\Assignment 1\\src\\hangman\\animals.txt");
  Scanner scanner = new Scanner(animals);
```



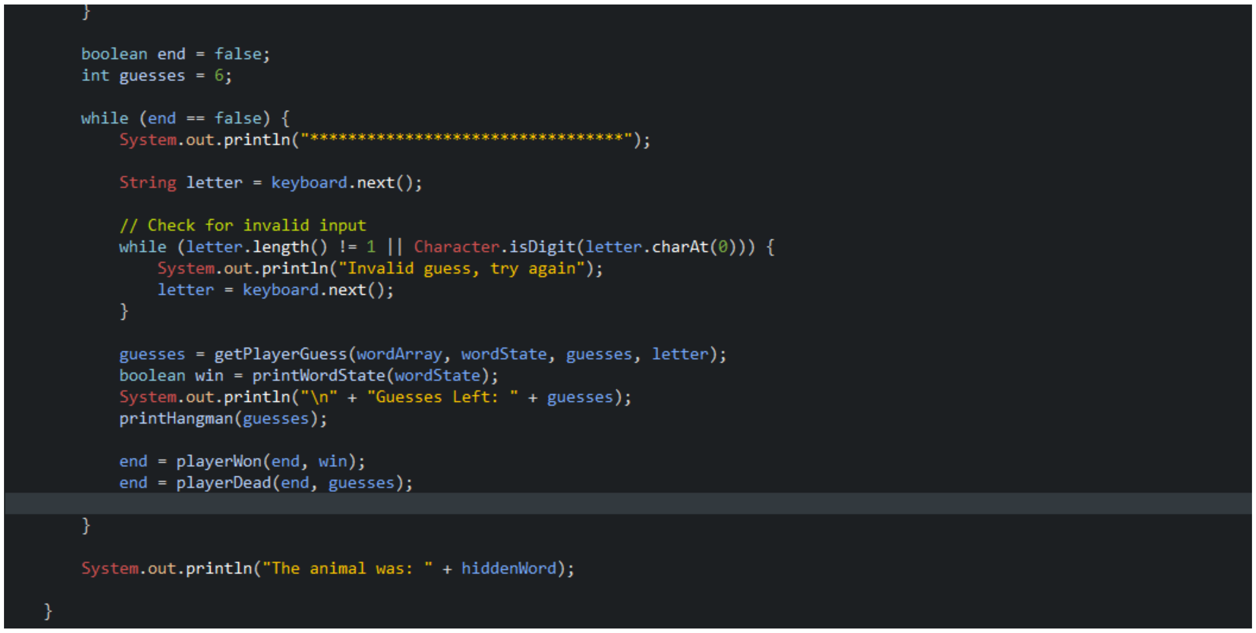
```
ArrayList<String> words = new ArrayList<>();

while(scanner.hasNext()) {
    words.add(scanner.nextLine());
}

String hiddenWord = words.get((int)(Math.random() * words.size()));
```

This game contains a main method and 5 other methods all of which play a key role and have their own functionality. Two different character arrays have been initialized, one to store the characters of the hidden animal word and one to store the characters of the generated animal's current state based off of user input.



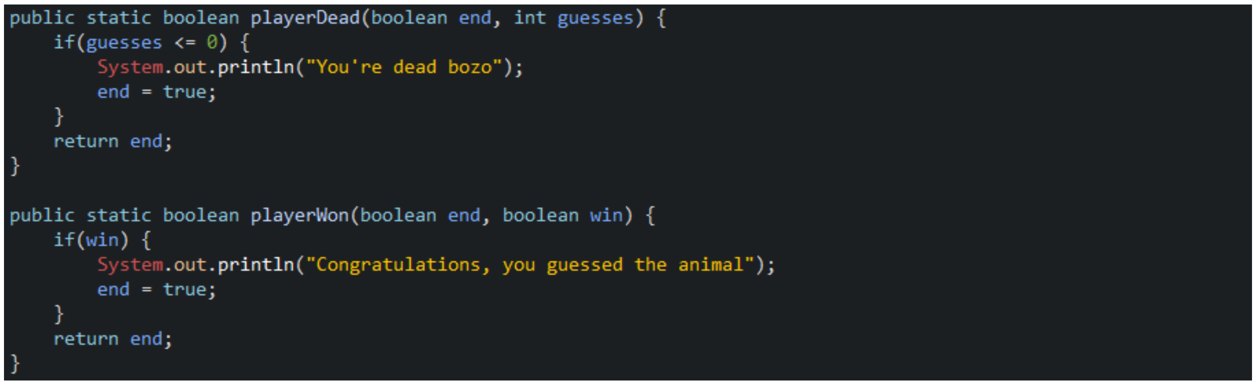
```java
char[] wordArray = hiddenWord.toCharArray();
char[] wordState = new char[wordArray.length];

for(int i = 0; i < wordArray.length; i++) {
    wordState[i] = '?';
}
```

The main method utilizes a while loop which runs off of the Boolean condition “end” initialized to false so that as long as the game has not ended, we do not break from this loop. In this while loop we create a string object to store the user’s guess and then we also check for validity of the user’s guess so long as their guess is not more than one letter nor a digit. We want to make sure the user knows how many guesses they have left so we make sure to display that in the terminal. The while loop also contains the calls of our other methods



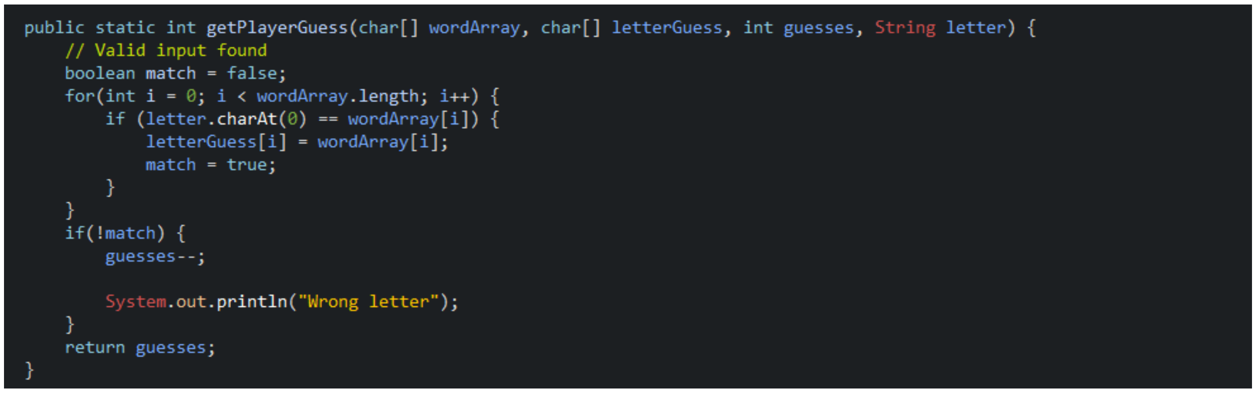
```java
        }

        boolean end = false;
        int guesses = 6;

        while (end == false) {
            System.out.println("*********************************");

            String letter = keyboard.next();

            // Check for invalid input
            while (letter.length() != 1 || Character.isDigit(letter.charAt(0))) {
                System.out.println("Invalid guess, try again");
                letter = keyboard.next();
            }

            guesses = getPlayerGuess(wordArray, wordState, guesses, letter);
            boolean win = printWordState(wordState);
            System.out.println("\n" + "Guesses Left: " + guesses);
            printHangman(guesses);

            end = playerWon(end, win);
            end = playerDead(end, guesses);

        }

        System.out.println("The animal was: " + hiddenWord);

    }
```

Now for the 5 other methods we have a method which returns a Boolean when the player has died, and another when the player has won. These are simple methods used to break from the game and return a message to the user for whether they have won or lost.



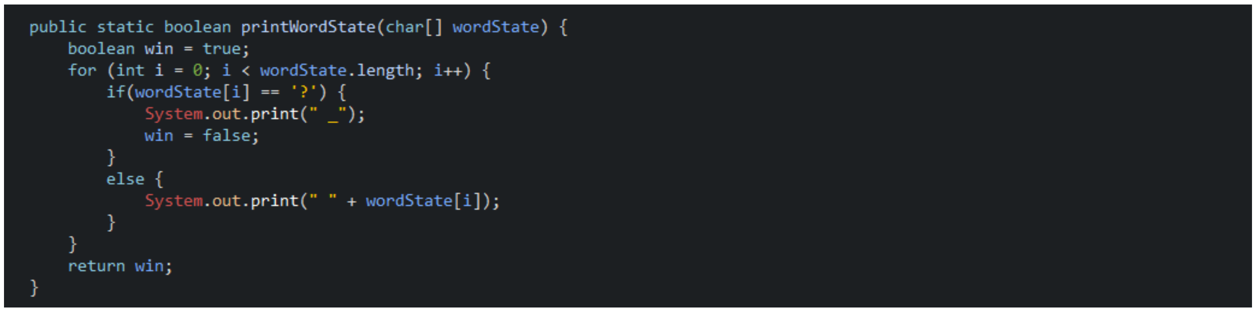
```java
public static boolean playerDead(boolean end, int guesses) {
    if(guesses <= 0) {
        System.out.println("You're dead bozo");
        end = true;
    }
    return end;
}

public static boolean playerWon(boolean end, boolean win) {
    if(win) {
        System.out.println("Congratulations, you guessed the animal");
        end = true;
    }
    return end;
}
```

Another method is used to get the player’s guess and store it in an array where all guesses taken can be found. This method then takes the player’s guess and compares for a matching character stored in our word array for our hidden animal. It then returns an integer value containing the player’s remaining guesses.



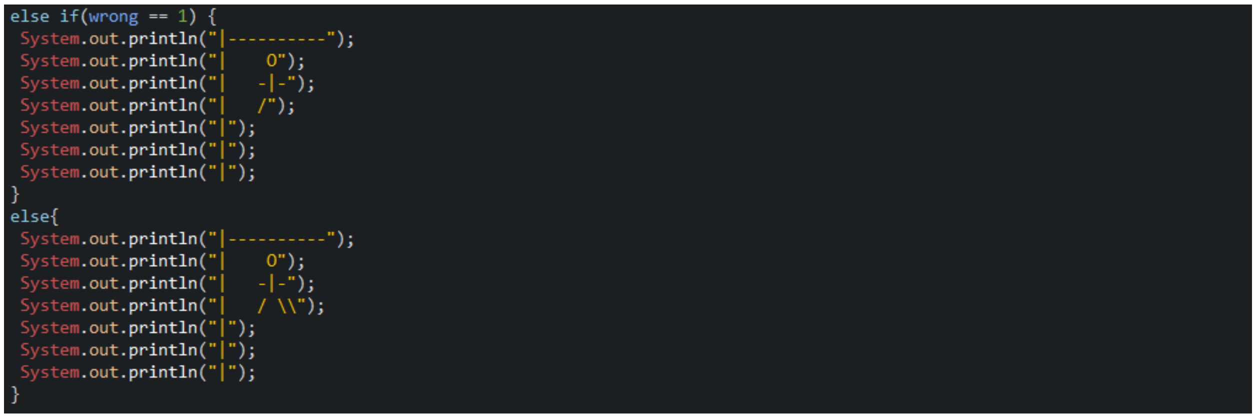
```java
public static int getPlayerGuess(char[] wordArray, char[] letterGuess, int guesses, String letter) {
    // Valid input found
    boolean match = false;
    for(int i = 0; i < wordArray.length; i++) {
        if (letter.charAt(0) == wordArray[i]) {
            letterGuess[i] = wordArray[i];
            match = true;
        }
    }
    if(!match) {
        guesses--;

        System.out.println("Wrong letter");
    }
    return guesses;
}
```

Additionally, we have a method which prints the state of the hidden word at the beginning of the game and after each of the player's guesses. If the letter is unknown to the user, the space is left empty, otherwise the letter is printed in the correct position matching the hidden word. This method returns a Boolean for whether the player has completely filled the word or not.



```java
public static boolean printWordState(char[] wordState) {
    boolean win = true;
    for (int i = 0; i < wordState.length; i++) {
        if(wordState[i] == '?') {
            System.out.print(" _");
            win = false;
        }
        else {
            System.out.print(" " + wordState[i]);
        }
    }
    return win;
}
```

The last method simply prints the hangman for each number of guesses that the player has gotten wrong (I'll only show the last two for context as it is a tedious method).



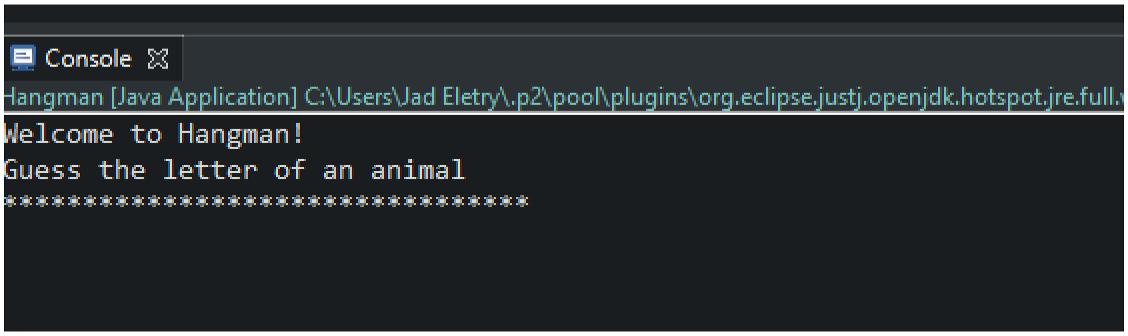
```java
else if(wrong == 1) {
 System.out.println("|----------");
 System.out.println("|    O");
 System.out.println("|   -|-");
 System.out.println("|   /");
 System.out.println("|");
 System.out.println("|");
 System.out.println("|");
}
else{
 System.out.println("|----------");
 System.out.println("|    O");
 System.out.println("|   -|-");
 System.out.println("|   / \\");
 System.out.println("|");
 System.out.println("|");
 System.out.println("|");
}
```

Challenges

I had some challenges come about when testing the printWordState method simply because I didn't know how to test a method which so frequently updates based on user input. So, I had to do a bit of research to try and figure out how exactly I wanted to test that method. Another complication while testing arose when I was going to test the printHangman method as I saw that it wouldn't be useful to test a method which simply returns a series of print statements and I thought it was going to be a bit tedious considering the amount of print statements needed to build the hanged man.

Sample Test Runs

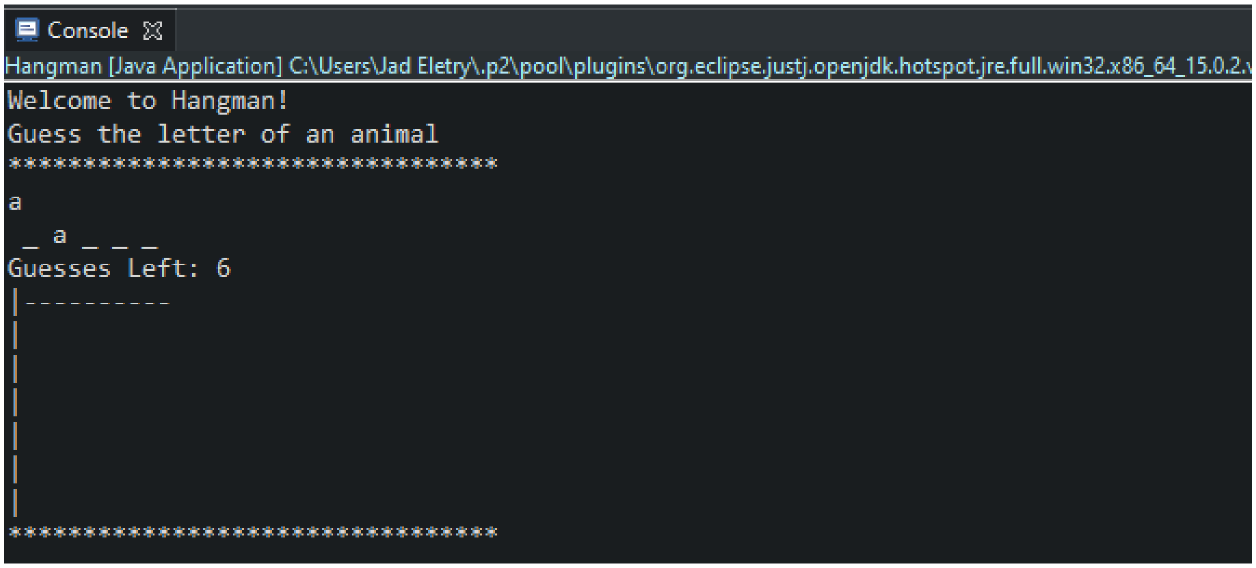
The game will start when you run the program and will be initialized in the terminal. From there you will be welcomed and prompted to guess a letter. The game is simple and easy to follow along, there shouldn't be any issues while doing so.



Console

```
Hangman [Java Application] C:\Users\Jad Eletry\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.
Welcome to Hangman!
Guess the letter of an animal
*********************************
```

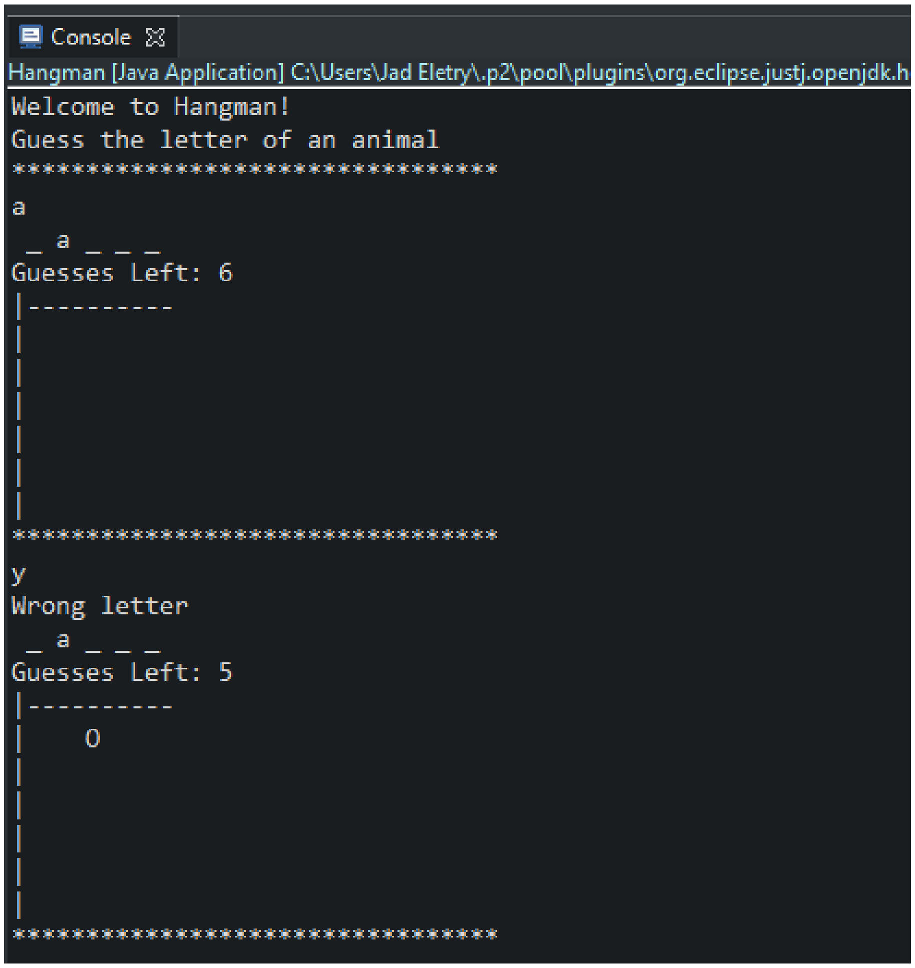
Now I will guess a letter (you can just click anywhere on the console and type).



Console

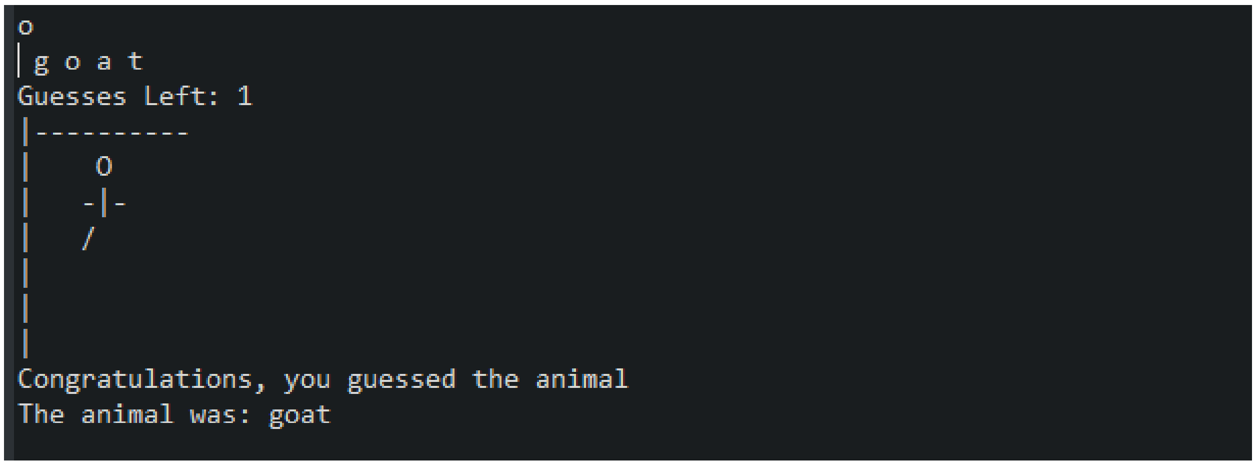
```
Hangman [Java Application] C:\Users\Jad Eletry\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.
Welcome to Hangman!
Guess the letter of an animal
*********************************
a
 _ a _ _ _
Guesses Left: 6
|----------
|
|
|
|
|
|
*********************************
```

I guessed the letter 'a' which was found to be a match, so I lost no guesses. But what if I guess wrong?



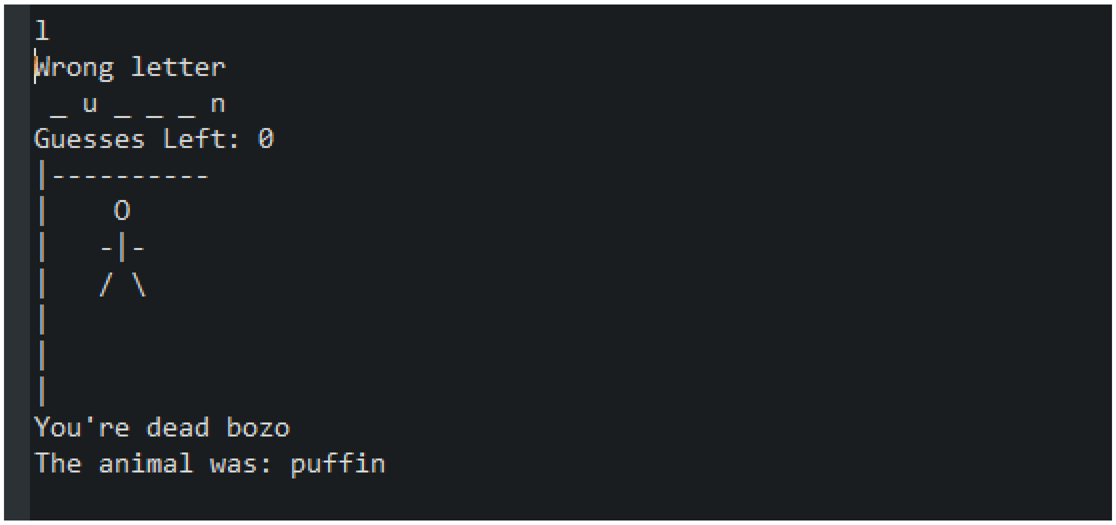
```
Console
Hangman [Java Application] C:\Users\Jad Eletry\.p2\pool\plugins\org.eclipse.justj.openjdk.h
Welcome to Hangman!
Guess the letter of an animal
*********************************
a
 _ a _ _ _
Guesses Left: 6
|----------
|
|
|
|
|
|
*********************************
y
Wrong letter
 _ a _ _ _
Guesses Left: 5
|----------
|    O
|
|
|
|
|
*********************************
```

As you can see, I guessed the letter ‘y’ which was wrong, and I lost a guess. You can also see the hanged man starting to get draw out. Now let me try and win the game.



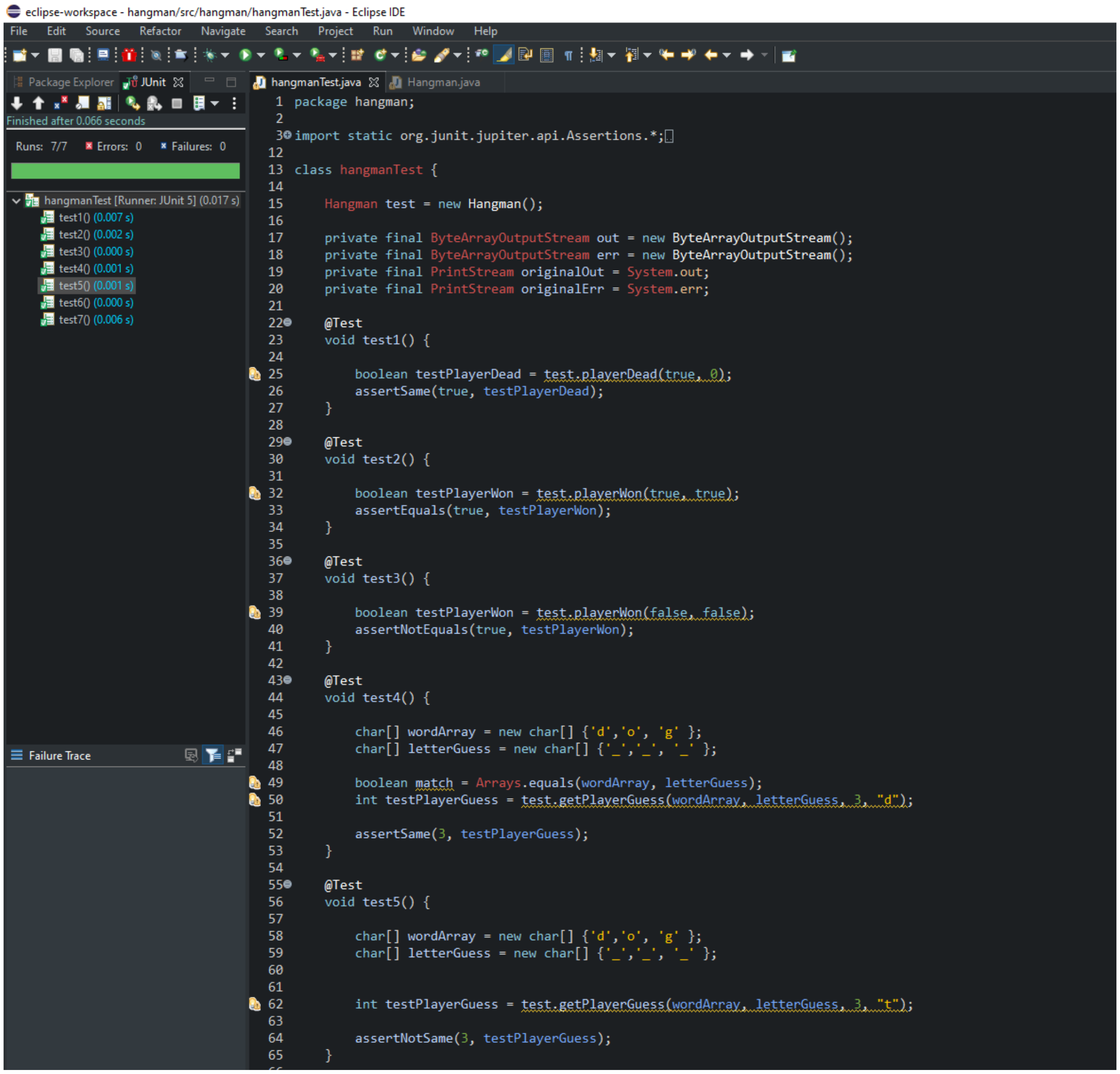
```
o
 g o a t
Guesses Left: 1
|----------
|    O
|   -|-
|   /
|
|
|
Congratulations, you guessed the animal
The animal was: goat
```

Now let’s try and lose the game.



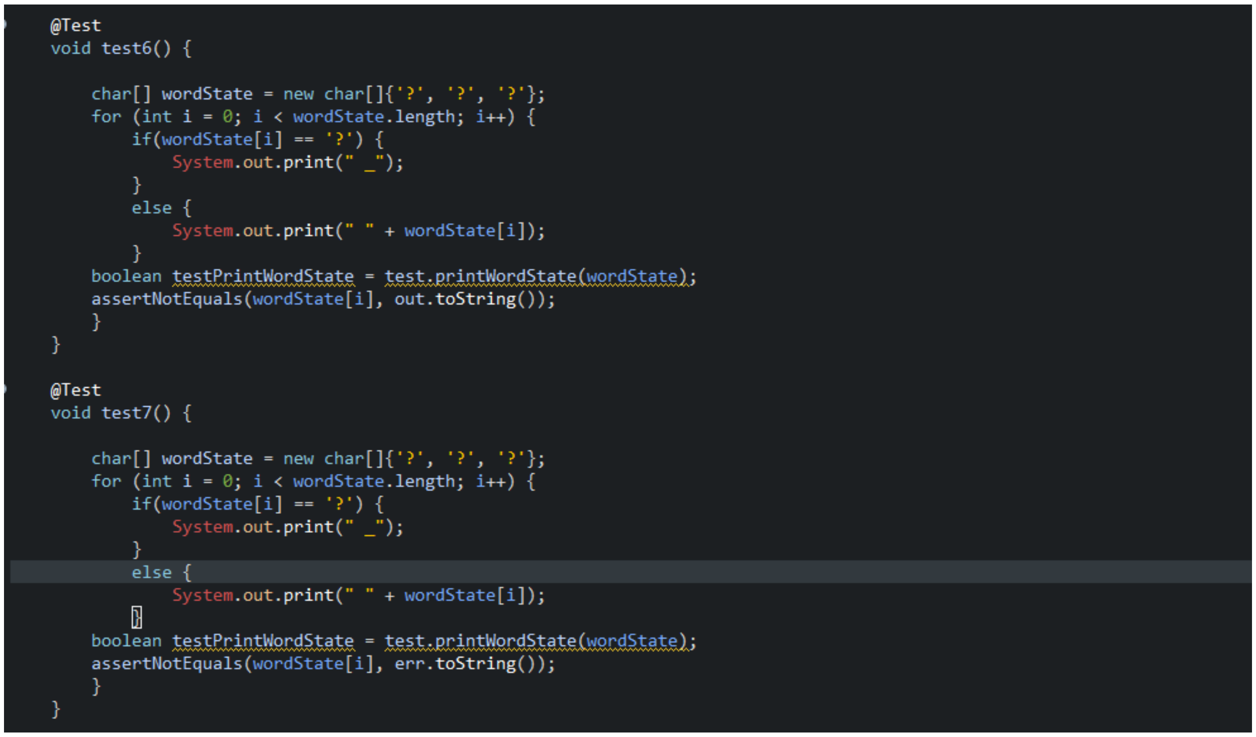
```
1
Wrong letter
 _ u _ _ _ n
Guesses Left: 0
|----------
|    O
|   -|-
|   / \
|
|
|
You're dead bozo
The animal was: puffin
```

As we can see, all components are fully functional. The test automation framework covers each method and tests their functionality. The image below displays this.



```
Finished after 0.066 seconds
Runs: 7/7    Errors: 0    Failures: 0

hangmanTest [Runner: JUnit 5] (0.017 s)
    test1() (0.007 s)
    test2() (0.002 s)
    test3() (0.000 s)
    test4() (0.001 s)
    test5() (0.001 s)
    test6() (0.000 s)
    test7() (0.006 s)
```

```java
package hangman;

import static org.junit.jupiter.api.Assertions.*;

class hangmanTest {

    Hangman test = new Hangman();

    private final ByteArrayOutputStream out = new ByteArrayOutputStream();
    private final ByteArrayOutputStream err = new ByteArrayOutputStream();
    private final PrintStream originalOut = System.out;
    private final PrintStream originalErr = System.err;

    @Test
    void test1() {

        boolean testPlayerDead = test.playerDead(true, 0);
        assertSame(true, testPlayerDead);
    }

    @Test
    void test2() {

        boolean testPlayerWon = test.playerWon(true, true);
        assertEquals(true, testPlayerWon);
    }

    @Test
    void test3() {

        boolean testPlayerWon = test.playerWon(false, false);
        assertNotEquals(true, testPlayerWon);
    }

    @Test
    void test4() {

        char[] wordArray = new char[] {'d','o', 'g' };
        char[] letterGuess = new char[] {'_','_', '_' };

        boolean match = Arrays.equals(wordArray, letterGuess);
        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "d");

        assertSame(3, testPlayerGuess);
    }

    @Test
    void test5() {

        char[] wordArray = new char[] {'d','o', 'g' };
        char[] letterGuess = new char[] {'_','_', '_' };


        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");

        assertNotSame(3, testPlayerGuess);
    }
```



```
@Test
void test6() {

    char[] wordState = new char[]{'?', '?', '?'};
    for (int i = 0; i < wordState.length; i++) {
        if(wordState[i] == '?') {
            System.out.print(" _");
        }
        else {
            System.out.print(" " + wordState[i]);
        }
    boolean testPrintWordState = test.printWordState(wordState);
    assertNotEquals(wordState[i], out.toString());
    }
}

@Test
void test7() {

    char[] wordState = new char[]{'?', '?', '?'};
    for (int i = 0; i < wordState.length; i++) {
        if(wordState[i] == '?') {
            System.out.print(" _");
        }
        else {
            System.out.print(" " + wordState[i]);
        }
    boolean testPrintWordState = test.printWordState(wordState);
    assertNotEquals(wordState[i], err.toString());
    }
}
```