

# 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.

```

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

```

    }

    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.

```

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.

```

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.

```

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).

```

else if(wrong == 1) {
    System.out.println("-----");
    System.out.println("|      0");
    System.out.println("|  -|-");
    System.out.println("|  /");
    System.out.println("|");
    System.out.println("|");
    System.out.println("|");
}
else{
    System.out.println("-----");
    System.out.println("|      0");
    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.

```

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).

[illegible]

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



```
1
Wrong letter
 _ u _ _ _ n
Guesses Left: 0
|-----
|      0
|      -|-
|      / \
|
|
|
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.

The screenshot displays the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. Below the menu is a toolbar with various icons for file operations and development tools. The left sidebar contains the Package Explorer, JUnit, and a console window showing the test results for HangmanTest. The main editor window displays the source code for HangmanTest.java.

**Test Results (JUnit):**

- 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)

**Source Code (HangmanTest.java):**

```

1 package hangman;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 class HangmanTest {
6
7     Hangman test = new Hangman();
8
9     private final ByteArrayOutputStream out = new ByteArrayOutputStream();
10    private final ByteArrayOutputStream err = new ByteArrayOutputStream();
11    private final PrintStream originalOut = System.out;
12    private final PrintStream originalErr = System.err;
13
14    @Test
15    void test1() {
16
17        boolean testPlayerDead = test.playerDead(true, 0);
18        assertEquals(true, testPlayerDead);
19    }
20
21    @Test
22    void test2() {
23
24        boolean testPlayerWon = test.playerWon(true, true);
25        assertEquals(true, testPlayerWon);
26    }
27
28    @Test
29    void test3() {
30
31        boolean testPlayerWon = test.playerWon(false, false);
32        assertEquals(true, testPlayerWon);
33    }
34
35    @Test
36    void test4() {
37
38        char[] wordArray = new char[] { 'd', 'o', 'g' };
39        char[] letterGuess = new char[] { '_', '_', '_' };
40
41        boolean match = Arrays.equals(wordArray, letterGuess);
42        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "d");
43
44        assertEquals(3, testPlayerGuess);
45    }
46
47    @Test
48    void test5() {
49
50        char[] wordArray = new char[] { 'd', 'o', 'g' };
51        char[] letterGuess = new char[] { '_', '_', '_' };
52
53        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
54
55        assertEquals(3, testPlayerGuess);
56    }
57
58    @Test
59    void test6() {
60
61        char[] wordArray = new char[] { 'd', 'o', 'g' };
62        char[] letterGuess = new char[] { '_', '_', '_' };
63
64        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
65
66        assertEquals(3, testPlayerGuess);
67    }
68
69    @Test
70    void test7() {
71
72        char[] wordArray = new char[] { 'd', 'o', 'g' };
73        char[] letterGuess = new char[] { '_', '_', '_' };
74
75        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
76
77        assertEquals(3, testPlayerGuess);
78    }
79
80    @Test
81    void test8() {
82
83        char[] wordArray = new char[] { 'd', 'o', 'g' };
84        char[] letterGuess = new char[] { '_', '_', '_' };
85
86        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
87
88        assertEquals(3, testPlayerGuess);
89    }
90
91    @Test
92    void test9() {
93
94        char[] wordArray = new char[] { 'd', 'o', 'g' };
95        char[] letterGuess = new char[] { '_', '_', '_' };
96
97        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
98
99        assertEquals(3, testPlayerGuess);
100    }
101
102    @Test
103    void test10() {
104
105        char[] wordArray = new char[] { 'd', 'o', 'g' };
106        char[] letterGuess = new char[] { '_', '_', '_' };
107
108        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
109
110        assertEquals(3, testPlayerGuess);
111    }
112
113    @Test
114    void test11() {
115
116        char[] wordArray = new char[] { 'd', 'o', 'g' };
117        char[] letterGuess = new char[] { '_', '_', '_' };
118
119        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
120
121        assertEquals(3, testPlayerGuess);
122    }
123
124    @Test
125    void test12() {
126
127        char[] wordArray = new char[] { 'd', 'o', 'g' };
128        char[] letterGuess = new char[] { '_', '_', '_' };
129
130        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
131
132        assertEquals(3, testPlayerGuess);
133    }
134
135    @Test
136    void test13() {
137
138        char[] wordArray = new char[] { 'd', 'o', 'g' };
139        char[] letterGuess = new char[] { '_', '_', '_' };
140
141        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
142
143        assertEquals(3, testPlayerGuess);
144    }
145
146    @Test
147    void test14() {
148
149        char[] wordArray = new char[] { 'd', 'o', 'g' };
150        char[] letterGuess = new char[] { '_', '_', '_' };
151
152        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
153
154        assertEquals(3, testPlayerGuess);
155    }
156
157    @Test
158    void test15() {
159
160        char[] wordArray = new char[] { 'd', 'o', 'g' };
161        char[] letterGuess = new char[] { '_', '_', '_' };
162
163        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
164
165        assertEquals(3, testPlayerGuess);
166    }
167
168    @Test
169    void test16() {
170
171        char[] wordArray = new char[] { 'd', 'o', 'g' };
172        char[] letterGuess = new char[] { '_', '_', '_' };
173
174        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
175
176        assertEquals(3, testPlayerGuess);
177    }
178
179    @Test
180    void test17() {
181
182        char[] wordArray = new char[] { 'd', 'o', 'g' };
183        char[] letterGuess = new char[] { '_', '_', '_' };
184
185        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
186
187        assertEquals(3, testPlayerGuess);
188    }
189
190    @Test
191    void test18() {
192
193        char[] wordArray = new char[] { 'd', 'o', 'g' };
194        char[] letterGuess = new char[] { '_', '_', '_' };
195
196        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
197
198        assertEquals(3, testPlayerGuess);
199    }
200
201    @Test
202    void test19() {
203
204        char[] wordArray = new char[] { 'd', 'o', 'g' };
205        char[] letterGuess = new char[] { '_', '_', '_' };
206
207        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
208
209        assertEquals(3, testPlayerGuess);
210    }
211
212    @Test
213    void test20() {
214
215        char[] wordArray = new char[] { 'd', 'o', 'g' };
216        char[] letterGuess = new char[] { '_', '_', '_' };
217
218        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
219
220        assertEquals(3, testPlayerGuess);
221    }
222
223    @Test
224    void test21() {
225
226        char[] wordArray = new char[] { 'd', 'o', 'g' };
227        char[] letterGuess = new char[] { '_', '_', '_' };
228
229        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
230
231        assertEquals(3, testPlayerGuess);
232    }
233
234    @Test
235    void test22() {
236
237        char[] wordArray = new char[] { 'd', 'o', 'g' };
238        char[] letterGuess = new char[] { '_', '_', '_' };
239
240        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
241
242        assertEquals(3, testPlayerGuess);
243    }
244
245    @Test
246    void test23() {
247
248        char[] wordArray = new char[] { 'd', 'o', 'g' };
249        char[] letterGuess = new char[] { '_', '_', '_' };
250
251        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
252
253        assertEquals(3, testPlayerGuess);
254    }
255
256    @Test
257    void test24() {
258
259        char[] wordArray = new char[] { 'd', 'o', 'g' };
260        char[] letterGuess = new char[] { '_', '_', '_' };
261
262        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
263
264        assertEquals(3, testPlayerGuess);
265    }
266
267    @Test
268    void test25() {
269
270        char[] wordArray = new char[] { 'd', 'o', 'g' };
271        char[] letterGuess = new char[] { '_', '_', '_' };
272
273        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
274
275        assertEquals(3, testPlayerGuess);
276    }
277
278    @Test
279    void test26() {
280
281        char[] wordArray = new char[] { 'd', 'o', 'g' };
282        char[] letterGuess = new char[] { '_', '_', '_' };
283
284        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
285
286        assertEquals(3, testPlayerGuess);
287    }
288
289    @Test
290    void test27() {
291
292        char[] wordArray = new char[] { 'd', 'o', 'g' };
293        char[] letterGuess = new char[] { '_', '_', '_' };
294
295        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
296
297        assertEquals(3, testPlayerGuess);
298    }
299
300    @Test
301    void test28() {
302
303        char[] wordArray = new char[] { 'd', 'o', 'g' };
304        char[] letterGuess = new char[] { '_', '_', '_' };
305
306        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
307
308        assertEquals(3, testPlayerGuess);
309    }
310
311    @Test
312    void test29() {
313
314        char[] wordArray = new char[] { 'd', 'o', 'g' };
315        char[] letterGuess = new char[] { '_', '_', '_' };
316
317        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
318
319        assertEquals(3, testPlayerGuess);
320    }
321
322    @Test
323    void test30() {
324
325        char[] wordArray = new char[] { 'd', 'o', 'g' };
326        char[] letterGuess = new char[] { '_', '_', '_' };
327
328        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
329
330        assertEquals(3, testPlayerGuess);
331    }
332
333    @Test
334    void test31() {
335
336        char[] wordArray = new char[] { 'd', 'o', 'g' };
337        char[] letterGuess = new char[] { '_', '_', '_' };
338
339        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
340
341        assertEquals(3, testPlayerGuess);
342    }
343
344    @Test
345    void test32() {
346
347        char[] wordArray = new char[] { 'd', 'o', 'g' };
348        char[] letterGuess = new char[] { '_', '_', '_' };
349
350        int testPlayerGuess = test.getPlayerGuess(wordArray, letterGuess, 3, "t");
351
352        assertEquals(3, testPlayerGuess);
353    }
354
355    @Test
356    void test33() {
357
358        char[] wordArray = new char[] { 'd', 'o', 'g' };
359        char[] letterGuess = new char[] { '_', '_', '_' };
360
361        int testPlayerGuess = test.getPlayerGuess(wordArray
```

```

@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);
    assertEquals(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);
    assertEquals(wordState[i], err.toString());
}

```