Part 1 functional Testing

For this project we have the game environment set up and have a test background as well. The only functionalities we have is moving the player with w, a, s, and d. In the table below, we tested each input variable as the keys were pressed to ensure it was the proper key. Also, we tested to see if the player did move a distance with each button pressed.

Table 1: The specification of equivalence partitioning

Partition ID	Input Variable	Valid Partition	Invalid Partition
1	w	yes	no
2	а	yes	no
3	s	yes	no
4	d	yes	no

Table 2: The test case specifications for each partition

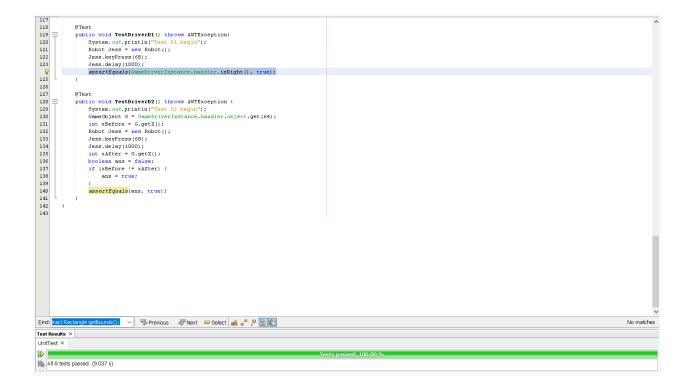
Test ID	Test Inputs	Expected Output	Partition ID Covered
1	w	Object moves up	1
2	а	Object moves left	2
3	S	Object moves down	3
4	d	Object moves right	4

JUnit implementation of the test cases implemented

```
7 🗇 import org.junit.After;
 <u>@</u>
      import org.junit.AfterClass;
 9
      import org.junit.Before;
 <u>@</u>
      import org.junit.BeforeClass;
11
      import org.junit.Test;
 Q
      import static org.junit.Assert.*;
13
14
      import MyGame.*;
15
      import java.awt.AWTException;
 <u>@</u>
      import java.awt.event.KeyEvent;
17
      import static junit.framework.Assert.assertEquals;
      import java.awt.event.KeyAdapter;
   └ import java.awt.Robot;
19
20
21 📮 / **
22
       * @author aaronramirez
23
   L | */
24
25
      public class UnitTest {
26
          Game GameDriverInstance;
27
          KeyInput MyKeyInput;
28
29
30
31
          @Before
32 🖃
          public void setUpClass() {
33
              System.out.println("Gamer test before: Before method test()");
              GameDriverInstance = new Game();
34
35
36
37
          @After
38
   public void tearDownClass() {
39
              System.out.println("Gamer test after: After method test()");
              GameDriverInstance = null;
40
41
42
43
          @Test
          public void TestDriverA1() throws AWTException(
44 -
45
              System.out.println("Test A1 begin");
46
              Robot Jess = new Robot();
```

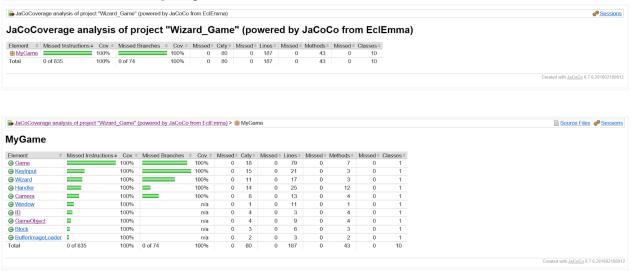
```
Robot Jess = new Robot();
46
47
              Jess.keyPress(65);
              Jess.delay(300);
48
              assertEquals (GameDriverInstance.handler.isLeft(), true);
49
50
51
52
          @Test
53
          public void TestDriverA2() throws AWTException {
              System.out.println("Test A2 begin");
54
              GameObject G = GameDriverInstance.handler.object.get(64);
55
              int xBefore = G.getX();
56
57
              Robot Jess = new Robot();
58
              Jess.keyPress(65);
59
              Jess.delay(1000);
              int xAfter = G.getX();
60
61
              boolean ans = false;
              if (xBefore != xAfter) {
62
63
                  ans = true;
64
65
              assertEquals(ans, true);
66
67
68
          @Test
          public void TestDriverW1() throws AWTException{
69
70
              System.out.println("Test W1 begin");
71
              Robot Jess = new Robot();
72
              Jess.keyPress(87);
73
              Jess.delay(1000);
74
              assertEquals(GameDriverInstance.handler.isUp(), true);
75
76
77
          @Test
78
   public void TestDriverW2() throws AWTException {
79
              System.out.println("Test W2 begin");
80
              GameObject G = GameDriverInstance.handler.object.get(64);
              int yBefore = G.getY();
81
82
              Robot Jess = new Robot();
83
              Jess.keyPress(87);
84
              Jess.delay(1000);
              int yAfter = G.getY();
85
```

```
85
               int yAfter = G.getY();
86
               boolean ans = false;
87
               if (yBefore != yAfter) {
                   ans = true;
88
89
               assertEquals (ans, true);
90
91
92
93
           @Test
94 🖃
           public void TestDriverS1() throws AWTException{
95
               System.out.println("Test S1 begin");
               Robot Jess = new Robot();
96
97
               Jess.keyPress(83);
98
               Jess.delay(1000);
               assertEquals(GameDriverInstance.handler.isDown(), true);
99
100
101
102
           @Test
103
           public void TestDriverS2() throws AWTException {
104
               System.out.println("Test S2 begin");
105
               GameObject G = GameDriverInstance.handler.object.get(64);
106
               int yBefore = G.getY();
               Robot Jess = new Robot();
107
108
               Jess.keyPress(83);
109
               Jess.delay(1000);
110
               int yAfter = G.getY();
111
               boolean ans = false;
112
               if (yBefore != yAfter) {
113
                   ans = true;
114
115
               assertEquals(ans, true);
116
117
118
           @Test
119 🖃
           public void TestDriverD1() throws AWTException{
120
               System.out.println("Test D1 begin");
121
               Robot Jess = new Robot();
122
               Jess.keyPress(68);
123
               Jess.delay(1000);
124
               assertEquals(GameDriverInstance.handler.isRight(), true);
```



Part 2 Structural Testing:

100% statement adequacy



Part 3: Contribution of each teammate:

Harsh: He did structural and functional testing for the W button. He went and tested the positions of the start and end points when the button is pressed. He also tested if the button is pressed it, the player character would be moved up. He also helped find the

test coverage tool JaCoCo. He double checked the JaCoCo testing and got the same results as Abbey.

Jess: She did structural and function testing for the S button. She went and tested the positions of the start and end points when the button is pressed. She also tested if the button is pressed it, the player character would be moved down. She did the partition ID table.

Aaron: He did structural and functional testing for the A button. He went and tested the positions of the start and end points when the button is pressed. He also tested if the button is pressed it, the player character would be moved left. He did the Test ID table with the testing.

Abbey: She did structural and function testing for the D button. She went and tested the positions of the start and end points when the button is pressed. She also tested if the button is pressed it, the player character would be moved right. She also ran the JaCoCo testing and showed the results.