

```

1 import static org.junit.Assert.*;
5
6 public class GameTest {
7
8     @Test
9     public void testMoveNorthFalse() {
10         //verify that if the current room does not have a North Door, you
        cannot move north
11         Room mockedRoom = Mockito.mock(Room.class);
12         Mockito.when(mockedRoom.getNorthDoor()).thenReturn(false);
13         Game g = new Game();
14         g.setCurrentRoom(mockedRoom);
15         assertFalse(g.moveNorth());
16
17     }
18
19     @Test
20     public void testMoveNorthTrue() {
21         //verify that if the current room has a north door, you can move north
22         Room mockedRoom = Mockito.mock(Room.class);
23         Mockito.when(mockedRoom.getNorthDoor()).thenReturn(true);
24         Game g = new Game();
25         g.setCurrentRoom(mockedRoom);
26         assertTrue(g.moveNorth());
27
28     }
29
30     @Test
31     public void testMoveSouthFalse() {
32         //verify that if the current room does not have a South Door, you
        cannot move south
33         Room mockedRoom = Mockito.mock(Room.class);
34         Mockito.when(mockedRoom.getSouthDoor()).thenReturn(false);
35         Game g = new Game();
36         g.setCurrentRoom(mockedRoom);
37         assertFalse(g.moveSouth());
38
39     }
40
41     @Test
42     public void testMoveSouthTrue() {
43         //verify that if the current room has a south Door, you cannot move
        south
44         Room mockedRoom = Mockito.mock(Room.class);
45         Mockito.when(mockedRoom.getSouthDoor()).thenReturn(true);
46         Game g = new Game();
47         g.setCurrentRoom(mockedRoom);
48         assertTrue(g.moveSouth());

```

```

49     }
50
51     @Test
52     public void testLookCream(){
53         //verify that if the room has cream in it, the game tells you that you
        have cream
54         Room mockedRoom = Mockito.mock(Room.class);
55         Mockito.when(mockedRoom.getObjectInRoom()).thenReturn(2);
56         Game g = new Game();
57         g.setCurrentRoom(mockedRoom);
58         assertEquals(g.look(), "You found some creamy cream!");
59     }
60
61     @Test
62     public void testLookSugar(){
63         //verify that if the room has sugar in it, the game tells you that you
        have sugar
64         Room mockedRoom = Mockito.mock(Room.class);
65         Mockito.when(mockedRoom.getObjectInRoom()).thenReturn(1);
66         Game g = new Game();
67         g.setCurrentRoom(mockedRoom);
68         assertEquals(g.look(), "You found some sweet sugar!");
69     }
70
71     @Test
72     public void testLookCoffee(){
73         //verify that if the room has coffee in it, the game tells you that
        you have coffee
74         Room mockedRoom = Mockito.mock(Room.class);
75         Mockito.when(mockedRoom.getObjectInRoom()).thenReturn(3);
76         Game g = new Game();
77         g.setCurrentRoom(mockedRoom);
78         assertEquals(g.look(), "You found some caffeinated coffee!");
79     }
80
81     @Test
82     public void testLookNothing(){
83         //verify that if the room has nothing in it, the game tells you that
        you have nothing
84         Room mockedRoom = Mockito.mock(Room.class);
85         Mockito.when(mockedRoom.getObjectInRoom()).thenReturn(0);
86         Game g = new Game();
87         g.setCurrentRoom(mockedRoom);
88         assertEquals(g.look(), "You don't see anything out of the ordinary.");
89     }
90
91
92 }

```

