```java
import static org.junit.Assert.*;

public class RoomTest {

    @Test
    public void testPopulateRoom0() {
        //check that all variables are set properly for room 0
        //make sure the north door, south door, cream coffee and sugar are in the correct states
        //since they are hardcoded in.
        Room x = new Room(0);
        assertTrue(x.northDoor);
        assertFalse(x.southDoor);
        assertTrue(x.cream);
        assertFalse(x.coffee);
        assertFalse(x.sugar);
    }
    @Test
    public void testPopulateRoom1() {
        //check that all variables are set properly for room 1
        //make sure the north door, south door, cream coffee and sugar are in the correct states
        //since they are hardcoded in.
        Room x = new Room(1);
        assertTrue(x.northDoor);
        assertTrue(x.southDoor);
        assertFalse(x.cream);
        assertTrue(x.coffee);
        assertFalse(x.sugar);
    }
    @Test
    public void testPopulateRoom2() {
        //check that all variables are set properly for room 2
        //make sure the north door, south door, cream coffee and sugar are in the correct states
        //since they are hardcoded in.
        Room x = new Room(2);
        assertTrue(x.northDoor);
        assertTrue(x.southDoor);
        assertFalse(x.cream);
        assertFalse(x.coffee);
        assertFalse(x.sugar);
    }
    @Test
    public void testPopulateRoom3() {
        //check that all variables are set properly for room 3
        //make sure the north door, south door, cream coffee and sugar are in the correct states
```

```
49          //since they are hardcoded in.
50          Room x = new Room(3);
51          assertTrue(x.northDoor);
52          assertTrue(x.southDoor);
53          assertFalse(x.cream);
54          assertFalse(x.coffee);
55          assertFalse(x.sugar);
56      }
57      @Test
58      public void testPopulateRoom4() {
59          //check that all variables are set properly for room 4
60          //make sure the north door, south door, cream coffee and sugar are in
    the correct states
61          //since they are hardcoded in.
62          Room x = new Room(4);
63          assertTrue(x.northDoor);
64          assertTrue(x.southDoor);
65          assertFalse(x.cream);
66          assertFalse(x.coffee);
67          assertFalse(x.sugar);
68      }
69      @Test
70      public void testPopulateRoom5() {
71          //check that all variables are set properly for room 5
72          //make sure the north door, south door, cream coffee and sugar are in
    the correct states
73          //since they are hardcoded in.
74          Room x = new Room(5);
75          assertFalse(x.northDoor);
76          assertTrue(x.southDoor);
77          assertFalse(x.cream);
78          assertFalse(x.coffee);
79          assertTrue(x.sugar);
80      }
81
82      @Test
83      public void testGetRoomDescription() {
84          //check that the Room Descriptions are properly set by creating each
    of the rooms, and verifying the room adjective matches
85          //the expected value array, called adjArray.  Also all of these
    adjectives describe out professor, Bill Laboon.
86          String[] adjArray={"Inspirational", "Cool-Dude","Chili-
    Pepper","Smart", "Fun", "Hilarious"};
87          for(int i=0;i<6;i++)
88          {
89              Room x = new Room(i);
90              assertEquals(x.roomAdj,adjArray[i]);
91          }
```

```
92      }
93
94      @Test
95      public void testGetRoomDescriptionInvalid() {
96          //Verify that if the room number is invalid, the error message is the
    room adjective.
97          Room x = new Room(6);
98          assertEquals(x.roomAdj, "Error: Invalid room number");
99      }
100
101     @Test
102     public void testGetObjDescription() {
103         //check that all variables are set properly for room 0
104         //make sure the north door, south door, cream coffee and sugar are in
    the correct states
105         //since they are hard-coded in.
106         String[] objArray={"a statue of Bill Laboon", "Amazon's best-seller,
    \"A Friendly Introduction to Software Testing\" by THE Bill Laboon",
107                 "an autographed photo of Bill Laboon",  "\"Hackin' Fellow\"
    on repeat 'cause it's such an amazing song", "a broken record","RentACat
    cats"};
108         for(int i=0;i<6;i++)
109         {
110             Room x = new Room(i);
111             assertEquals(x.objAdj,objArray[i]);
112         }
113     }
114
115     @Test
116     public void testGetObjectInRoomCream() {
117         //verify that the getObject in room returns the correct int for room
    0
118         //this room has cream, so verify it returns properly
119         Room x = new Room(0);
120         assertEquals(x.getObjectInRoom(),2);
121
122     }
123     @Test
124     public void testGetObjectInRoomCoffee() {
125         //verify that the getObject in room returns the correct int for room
    1
126         //this room has coffee, so verify it returns properly
127         Room x = new Room(1);
128         assertEquals(x.getObjectInRoom(),3);
129
130     }
131     @Test
132     public void testGetObjectInRoomSugar() {
```

```
133         //verify that the getObject in room returns the correct int for room
    5
134         //this room has sugar, so verify it returns properly
135         Room x = new Room(5);
136         assertEquals(x.getObjectInRoom(),1);
137 
138     }
139     @Test
140     public void testGetObjectInRoomFail() {
141         //verify that the getObject in room returns the correct int for room
    3
142         //this room has nothing, so verify it returns properly
143         Room x = new Room(3);
144         assertEquals(x.getObjectInRoom(),0);
145 
146     }
147 
148     @Test
149     public void testGetNorthDoorTrue(){
150         //verify that the setters and getters for North Door work
151         //we test room 0, which is a valid room
152         Room x = new Room(0);
153         x.setNorthDoor();
154         assertTrue(x.getNorthDoor());
155     }
156 
157     @Test
158     public void testGetNorthDoorFalse(){
159         //verify that the setters and getters for North Door work, if north
    door is set to false.
160         //we test room 0, which is a valid room
161         Room x = new Room(5);
162         assertFalse(x.getNorthDoor());
163     }
164 
165     @Test
166     public void testGetSouthDoorTrue(){
167         //verify that the setters and getters for South Door work
168         //we test room 0, which is a valid room
169         Room x = new Room(0);
170         x.setSouthDoor();
171         assertTrue(x.getSouthDoor());
172     }
173 }
174 
```