```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
        //since they are hardcoded in.
        Room x = new Room(3);
        assertTrue(x.northDoor);
        assertTrue(x.southDoor);
        assertFalse(x.cream);
        assertFalse(x.coffee);
        assertFalse(x.sugar);
    }
    @Test
    public void testPopulateRoom4() {
        //check that all variables are set properly for room 4
        //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(4);
        assertTrue(x.northDoor);
        assertTrue(x.southDoor);
        assertFalse(x.cream);
        assertFalse(x.coffee);
        assertFalse(x.sugar);
    }
    @Test
    public void testPopulateRoom5() {
        //check that all variables are set properly for room 5
        //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(5);
        assertFalse(x.northDoor);
        assertTrue(x.southDoor);
        assertFalse(x.cream);
        assertFalse(x.coffee);
        assertTrue(x.sugar);
    }

    @Test
    public void testGetRoomDescription() {
        //check that the Room Descriptions are properly set by creating each of the rooms, and verifying the room adjective matches
        //the expected value array, called adjArray.  Also all of these adjectives describe out professor, Bill Laboon.
        String[] adjArray={"Inspirational", "Cool-Dude","Chili-Pepper","Smart", "Fun", "Hilarious"};
        for(int i=0;i<6;i++)
        {
            Room x = new Room(i);
            assertEquals(x.roomAdj,adjArray[i]);
        }
    }

    @Test
    public void testGetRoomDescriptionInvalid() {
        //Verify that if the room number is invalid, the error message is the room adjective.
        Room x = new Room(6);
        assertEquals(x.roomAdj, "Error: Invalid room number");
    }

    @Test
    public void testGetObjDescription() {
        //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 hard-coded in.
        String[] objArray={"a statue of Bill Laboon", "Amazon's best-seller, \"A Friendly Introduction to Software Testing\" by THE Bill Laboon",
                "an autographed photo of Bill Laboon",  "\"Hackin' Fellow\" on repeat 'cause it's such an amazing song", "a broken record","RentACat cats"};
        for(int i=0;i<6;i++)
        {
            Room x = new Room(i);
            assertEquals(x.objAdj,objArray[i]);
        }
    }

    @Test
    public void testGetObjectInRoomCream() {
        //verify that the getObject in room returns the correct int for room 0
        //this room has cream, so verify it returns properly
        Room x = new Room(0);
        assertEquals(x.getObjectInRoom(),2);

    }
    @Test
    public void testGetObjectInRoomCoffee() {
        //verify that the getObject in room returns the correct int for room 1
        //this room has coffee, so verify it returns properly
        Room x = new Room(1);
        assertEquals(x.getObjectInRoom(),3);

    }
    @Test
    public void testGetObjectInRoomSugar() {
```

```java
        //verify that the getObject in room returns the correct int for room 5
        //this room has sugar, so verify it returns properly
        Room x = new Room(5);
        assertEquals(x.getObjectInRoom(),1);

    }
    @Test
    public void testGetObjectInRoomFail() {
        //verify that the getObject in room returns the correct int for room 3
        //this room has nothing, so verify it returns properly
        Room x = new Room(3);
        assertEquals(x.getObjectInRoom(),0);

    }

    @Test
    public void testGetNorthDoorTrue(){
        //verify that the setters and getters for North Door work
        //we test room 0, which is a valid room
        Room x = new Room(0);
        x.setNorthDoor();
        assertTrue(x.getNorthDoor());
    }

    @Test
    public void testGetNorthDoorFalse(){
        //verify that the setters and getters for North Door work, if north door is set to false.
        //we test room 0, which is a valid room
        Room x = new Room(5);
        assertFalse(x.getNorthDoor());
    }

    @Test
    public void testGetSouthDoorTrue(){
        //verify that the setters and getters for South Door work
        //we test room 0, which is a valid room
        Room x = new Room(0);
        x.setSouthDoor();
        assertTrue(x.getSouthDoor());
    }
}
```