```
import java.util.*;
/**
 * This class builds rooms for the Game.
 *
 * @author Ben Parsell
 * @version 1.0.0 (11/19/2015)
 */
public class Room
{
    /** Describes the room **/
    private String roomDescription;

    /** Item in the room **/
    private Item optionalItem;

    /** ArrayList for rooms **/
    private HashMap <String, Room> neighbors;

    /**************************************************
     * Default Constructor for Room
     *
     * @param pDescription variable to describe room
     **************************************************/
    public Room(String pDescription) {
        // Initialize instance variable
        this.roomDescription = pDescription;

        // Creates hashmap for neighbors
        neighbors = new HashMap<String, Room>();
    }


    /**************************************************
     * Overloaded Constuctor
     *
     * @param pDescription variable to describe room
     * @param pItem variable for item in the room
     **************************************************/
    public Room(String pDescription, Item pItem) {
        // Initialize instance variables
        this.roomDescription = pDescription;
        this.optionalItem = pItem;

        // Creates hashmap for neighbors
        neighbors = new HashMap <String, Room> ();
    }


    /**************************************************
```

```
50      * Method to add an item into the game
51      *
52      * @param i variable for the item in the room
53      ***************************************************/
54     public void addItem(Item i) {
55         this.optionalItem = i;
56     }
57
58
59     /***************************************************
60      * Method to check if player has an item
61      *
62      * @return returns true of false depending on null
63      * value
64      ***************************************************/
65     public boolean hasItem() {
66         // Check for if room has an item
67         if(optionalItem != null) {
68             return true;
69         } else {
70             return false;
71         }
72     }
73
74
75     /***************************************************
76      * Method to get the description of the room
77      *
78      * @return roomDescription describes room
79      ***************************************************/
80     public String getDescription() {
81         return roomDescription;
82     }
83
84     /***************************************************
85      * Method to get an item in the Room
86      *
87      * @return optionalItem item within the room
88      ***************************************************/
89     public Item getItem() {
90         return optionalItem;
91     }
92
93
94     /***************************************************
95      * Method to view neighboring rooms
96      *
97      * @return value for neighboring rooms
98      ***************************************************/
```

```
99      public HashMap getNeighbors() {
100         return neighbors;
101     }
102
103
104     /**************************************************
105      * Method to add a neighboring room
106      *
107      * @param direction value for direction to move
108      * @param r value for room name
109      *************************************************/
110     public void addNeighbor(String direction, Room r) {
111         // Adds neighbor to room r, in that direction
112         this.neighbors.put(direction, r);
113     }
114
115
116     /**************************************************
117      * Finds neighbor in given direction
118      *
119      * @param direction value for direction to move
120      *************************************************/
121     public Room getNeighbor(String direction) {
122         // Create new room to find neighbor
123         Room neighbor = this.neighbors.get(direction);
124         return neighbor;
125     }
126
127
128     /**************************************************
129      * Method removes an item from the room
130      *
131      * @return optionalItem item within the room
132      *************************************************/
133     public Item removeItem() {
134         // Set item in room to null, to delete
135         this.optionalItem = null;
136         return optionalItem;
137     }
138
139
140     /**************************************************
141      * Method to retrieve the full description of a room
142      *
143      * @return longDescription value for well-described
144      * room information
145      *************************************************/
146     public String getLongDescription() {
147         // Check to see if room has an item, then return text
```

```
148         if(hasItem()) {
149             String longDescription = "You are currently in " + roomDescri
    ption + ". You see " + optionalItem.getDescription() + ".";
150             return longDescription + "\n";
151         }
152 
153         // If no item, gives room description without item.
154         else {
155             String longDescription = "You are in " + roomDescription + ".
    ";
156             return longDescription + "\n";
157         }
158     }
159 }
160 
```