Hotel - Array of objects example

The basic 'array of string' Hotel program can be improved using an array of objects. **Room.java** specifies a class (a template for an object). The program creates the object for each room with `new Room();` Each created object is self contained with its own variables and its own methods (function code which can change the values in its own variables). We can write code for objects so that the object (with its variables) is protected from outside interference apart from changes we have allowed with the object via the methods. If we know the object is protected then we can test that the object works properly and then have confidence that any changes to the outside program will not compromise the object. We could also re-use the same object in different programs and know it is going work as expected because it is protected.

Main.java

```
package arrayobjects;

import java.util.*;

public class HotelObjectsExample {

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        Room[] myHotel = new Room[4];
        myHotel[0] = new Room();    // 4 lines could go in loop
        myHotel[1] = new Room();
        myHotel[2] = new Room();
        myHotel[3] = new Room();

        String roomName;
        int roomNum = 0;

        initialise(myHotel);

        while (roomNum < 4) {
           for (int x = 0; x < 4; x++ )
           if (myHotel[x].mainName.equals("e"))
                  System.out.println("room " + x + " is empty");

           System.out.println("Enter room number (0-3) or 4 to stop:");
                                            //error with 4
           roomNum = input.nextInt();
           System.out.println("Enter name for room " + roomNum + " :");
           roomName = input.next();
            myHotel[roomNum].mainName = roomName ;
             // myHotel[roomNum].setName(roomName);
           for (int x = 0; x < 4; x++) {
             System.out.println("room " + x + " occupied by " + myHotel[x].mainName);
             // System.out.println("room " + x + " occupied by " + myHotel[x].getName());
           }
        }
    }
        private static void initialise( Room hotelRef[] ) {
         for (int x = 0; x < 4; x++ ) hotelRef[x].mainName = "e";
          System.out.println( "initilise ");
        }
}
```

Then create your class Room.java (In the same project create this with File – NewFile - JavaClass and call it Room , and paste this in as the public class Room.

```
public class Room {

     String mainName;
    //private String mainName;
    int guestsInRoom;

    public Room() {
        mainName = "k";
        System.out.println("made a room ");
    }

    public void setName(String aName) {
        System.out.println("add name class method ");
        mainName = aName;
    }

    public String getName() {
        return mainName;
    }
}
```

Initially the variables in **Room.java** will be public, and accessed directly from **Main.java**. In several cases in the following code, an instruction in **Main.java** directly accesses the object's variables. If these are commented out, and the following instruction uncommented, then the object will be accessed via methods (which is a safer way to write code). The object variables will be hidden, with access via methods, and a constructor will replace 'initialise'.

Create your own queue object (and queue methods)

```
//MyQueueObject.java
package myqueueobject;

import java.util.*;

public class MyQueueObject {

    /**
     * @param args the command line arguments
```