**Project - 1**

### Project Outcomes:

### Develop a Java program that uses:

* decision constructs
* looping constructs
* basic operations on an Array of objects (find, change, access all elements)
* more than one class and has multiple objects
* Object Oriented Programming features

### Project Requirements:

1. Develop a simple **Hotel** program. We will have two classes, a **Hotel** class representing an individual hotel and a **Room** class. The **Hotel** class will contain several **Room** objects and will have several operations. We will also have a driver program to test the **Hotel** class.
2. Build a **Hotel** class that will store information about a Hotel. It will include a **name** and **location**. It should also include an **Array** of instances of class **Room** to hold information about each room. It will also have an int called **occupiedCnt** that keeps track of how many rooms in the hotel are occupied.
3. Build a **HotelTest** class to test your application. Your test class should **not** require any interaction with the user. It should verify the correct operation of the constructor and all public methods in the **Hotel** class. Create at least 5 rooms.

**Specific Requirements for the Hotel Class:**

1. The Hotel class has two constructors
   1. A default constructors that sets the Array to a size of 10 and all instance fields to a default value.  We will not be using this constructor but you should almost always have one.
   2. The **Hotel** class parameterized constructor will read in the hotel name and location from hard-coded values in the tester class, such as Beach Marriot Pensacola, it will also assign numOfRooms to zero.  numOfRooms indicates how many rooms are in the hotel. It will create a 10 element array.
2. The **Hotel** will have an **addRoom** method that will create each room with the required information: room number, bed type, smoking/non-smoking, and the room rate. Create at least 5 rooms with different characteristics. Each room will also have a boolean field called **occupied** attribute that will be set to false when the room is created. Don't forget to increment the numOfRooms instance variable.  Example values for the rooms are:

101 queen s 100  
102 king n 110  
103 king n 88  
104 twin s 100  
105 queen n 99

1. The UML class diagram for the Hotel class will look like this:

|  |
| --- |
| Hotel |
| theRooms: Array Room[] name: String location: String occupiedCnt: int  numOfRooms: int |
| Hotel()  Hotel(String,String) isFull() : boolean  isEmpty() : boolean addRoom(int,String,char,double) addReservation(String,char,String)  cancelReservation(String) findReservation(String): int  printReservationList()  getDailySales() : double occupancyPercentage() : double  toString():String  Access and mutator methods for name and location. |

1. **isFull() –** returns a boolean that is true if all the rooms in the hotel are occupied.
2. **isEmpty() –** returns a boolean that is true if all the rooms in the hotel are unoccupied.
3. The **addReservation()** method takes three parameters: the occupant’s name (String), smoking or non-smoking request (char), and the requested bed type (String). When this method is called, the hotel will search the list of its rooms for one that matches the bed type and smoking/non-smoking attributes. If an unoccupied room with the correct attributes is found, the renter's name will be set and the **occupied** attribute will be set to true. In either case a message will be printed that will state whether or not the reservation was made.
4. When the **cancelReservation()** method executes, the hotel will search for the name of the visitor in each room. If it is found, the occupied attribute will be set to false. In either case a message will state whether or not the reservation was cancelled. This method calls the private utility method **findReservation()**to scan the list of rooms looking for a guest by name. It will return the index of the room in the **Array** of rooms or **NOT\_FOUND** if the room is not found, which will be declared as:

**private static final int NOT\_FOUND = -1**;

1. **findReservation()** will take in a String representing the occupant’s name and search the occupied rooms for a reservation with that person’s name. It will return the index of the room or **NOT\_FOUND** if not found.
2. **printReservationList()** will scan through all the rooms and display all details for only those rooms that are occupied. For example:

Room Number: 102  
Occupant name: Pinto  
Smoking room: n  
Bed Type: king  
Rate: 110.0  
  
Room Number: 103  
Occupant name: Wilson  
Smoking room: n  
Bed Type: king  
Rate: 88.0

1. **getDailySales()** will scan the room list, adding up the dollar amounts of the room rates of all occupied rooms only.
2. **occupancyPercentage()** will divide occupiedCnt by the total number of rooms to provide an occupancy percentage.
3. **toString() –** returns a nicely formatted string giving hotel and room details (by calling the **toString()** in the **Room** class) for all the rooms in the hotel. For example:

Hotel Name: Beach Marriot  
Number of Rooms: 5  
Number of Occupied Rooms: 1

Room Details are:

Room Number: 101  
Occupant name: Not Occupied   
Smoking room: s  
Bed Type: queen  
Rate: 100.0  
  
Room Number: 102  
Occupant name: Coffey  
Smoking room: n  
Bed Type: king  
Rate: 110.0  
  
Room Number: 103  
Occupant name: Wilson  
Smoking room: n  
Bed Type: king  
Rate: 88.0  
  
Room Number: 104  
Occupant name: Not Occupied  
Smoking room: s  
Bed Type: twin  
Rate: 100.0  
  
Room Number: 105  
Occupant name: Not Occupied  
Smoking room: n  
Bed Type: queen  
Rate: 99.0

1. The **Room** class diagram will look like this:

|  |
| --- |
| **Room** |
| roomNum: int bedType: String rate: double occupantName: String smoking: char occupied: boolean |
| Room()  Room(int,String,char,double) getBedType(): String getSmoking(): char         getRoomNum(): int  getRoomRate(): double  getOccupant(): String setOccupied(boolean) setOccupant(String)  setRoomNum(int)  setBedType(String)  setRate(double)  setSmoking(char)  isOccupied(): boolean toString(): String |

1. The constructor for a **Room** takes an int (room number), String (bed type), char (s or n for smoking or non-smoking)), and a double (room rate).
2. **isOccupied()** method returns true if the room is occupied, false otherwise.
3. **toString()** provides all the details of a room - room number, name of guest(if occupied) , bed type, smoking/non-smoking, rental rate. This should all be formatted nicely with one attribute on each line using the '\n' escape character. See example above.
4. Several accessor and mutator methods for the **Room** class.

**Submission Requirements:**

1. You should have at least three files for this assignment.
   1. Room.java
   2. Hotel.java
   3. HotelTester.java
2. Remember to compile and run your program one last time before you submit it.

---------------------------------------------------------------------------------------------------------------------