Short Answer:

Answer the following questions with complete sentences in your own words.
You are encouraged to conduct your own research online or through other methods before answering the questions. If you research online, please consult multiple sources before you write down your answers. You are expected to be able to explain your answers in detail (Provide examples to each question).

1. What is JDBC and How does it work?
2. What are the steps to connect to a database via JDBC?
3. What are JDBC API Components? (Interfaces and Classes)
4. What are JDBC statements? List all types of JDBC statements and their usage.
5. What are the differences between execute, executeQuery, and executeUpdate?
6. What is the role of the JDBC DriverManager class?
7. What is Transaction Management?
8. What is the CallableStatement interface in JDBC? Give an example.

Coding Questions:

Write code in Java to solve following problems. Please write your own answers. You are highly encouraged to present more than one way to answer the questions. Please follow best practice when you write the code so that it would be easily readable, maintainable, and efficient. Clearly state your assumptions if you have any. You may discuss with others on the questions, but please write your own code.

1. Implement the following operations as a single application.
   - Insert
   - Delete
   - Update
   - Display all records
   - Get

You have to display a menu to the customer which allows he/she to choose an operation. Depending on the customer choice, you have to run the corresponding methods.
[Insert a new record in database table, update the existing record, delete etc on student object]

2. Write a java program that takes tab separated data (one record per line) from a text file and inserts them into database
Eg. A file looks like as follows: (Each column is separated a tab)

```
PP 2222 19 BS 2001-12-07

II 3333 20 Master 2002-12-07
```

BB 5555 11 PHD 2003-12-07

ZZ 1111 44 Master 2004-12-07

SS 2222 11 PHD 2005-12-07
QQ 3333 88 BS 2006-12-07

DD 4444 99 PHD 2007-12-07

3. Hotel Project

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. You must use an Array, you cannot use an ArrayList. Set the Array to hold 10 Room objects.

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
```

3. 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.
```

4. **`isFull()`** – returns a boolean that is true if all the rooms in the hotel are occupied. 5.

`isEmpty()` – returns a boolean that is true if all the rooms in the hotel are unoccupied.

6. 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.

7. 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;
8. **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.
9. **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
```

10. **getDailySales()** will scan the room list, adding up the dollar amounts of the room rates of all occupied rooms only.
11. **occupancyPercentage()** will divide occupiedCnt by the total number of rooms to provide an occupancy percentage.
12. **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
```

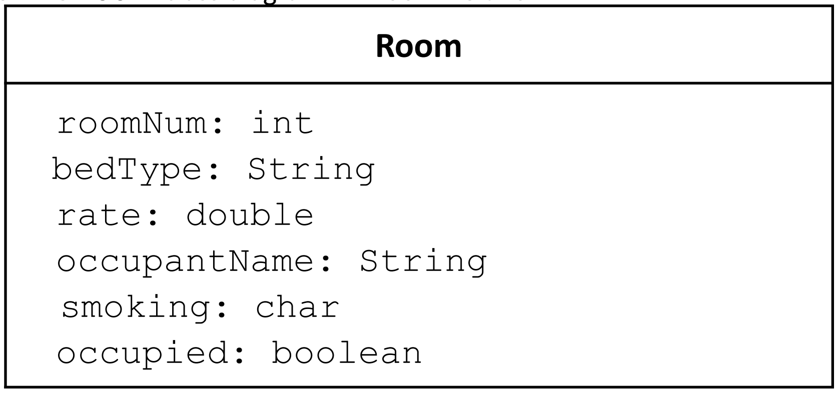
```
Smoking room: s
Bed Type: queen
Rate: 100.0
Room Number: 102
Occupant name: Coffey
Smoking room: n
Bed Type: king
Rate: 110.0

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
```

13. 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. Follow the submission requirements posted.
2. You should submit following files for this assignment.
    1. All the sql files.
    2. Room.java
    3. Hotel.java
    4. HotelTester.java