**Introduction to JDBC**

JDBC is an Application Programming Interface(API) which describes how a client may access a database. Here this program provides you the basic knowledge about SELECT statement in Oracle. The SELECT statement is used to retrieve the records from the database based on the client's needs. The retrieved data will be stored in a ResultSet and this ResultSet is used to display those selected records. Here, use a SELECT statement to display the records from the ItemType table.

**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**

Create a class **ItemType** with the following private attributes

|  |  |
| --- | --- |
| **Attribute** | **Datatype** |
| id | Long |
| name | String |
| deposit | Double |
| costPerDay | Double |

Generate appropriate **Getters**and **Setters**  
Generate **default**and **parameterized constructors**  
The parameterized constructor should be in the following format **ItemType(Long id, String name, Double deposit, Double costPerDay)**  
  
Create a class **ItemTypeDAO** with the following methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| public List<ItemType> getAllItemTypes() | This function finds all records of the ItemType table. It returns the ItemType objects in a List. |

Create a driver class **Main** to display the list of objects in the following format.

NOTE : Use **System.out.format("%-5s %-15s %-10s %s\n","ID","Name","Deposit","Cost per day")** to display ItemType details.  
  
**Table Properties:**

CREATE TABLE item\_type(  
id number(19) NOT NULL,  
name varchar2(255) NOT NULL,  
deposit BINARY\_DOUBLE NOT NULL,  
cost\_per\_day BINARY\_DOUBLE NOT NULL,  
PRIMARY KEY (id));

CREATE SEQUENCE item\_type\_seq START WITH 1 INCREMENT BY 1;

**Use the following code snippet to establish DBConnection:**

import java.util.ResourceBundle;

ResourceBundle rb = ResourceBundle.getBundle("oracle");

String url = rb.getString("db.url");

String username = rb.getString("db.username");

String password = rb.getString("db.password");

**oracle.properties:**

db.url = jdbc:oracle:thin:@localhost:1521:xe  
db.username = root  
db.password = student

**Download the oracle jar file in the below link.**  
[Oracle jar](http://hcl.e-box.co.in/uploads/ojdbc6.jar.zip)

**[All text in bold corresponds to the input and rest corresponds to the output]**

**Sample Input and Output:**

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Name | Deposit | Cost per day |
| 1 | Food | 50000.0 | 10000.0 |
| 2 | Electronics | 85000.0 | 15000.0 |
| 3 | Fashion | 36000.0 | 8000.0 |
| 4 | Grooming | 15000.0 | 5000.0 |
| 5 | Books | 20000.0 | 7500.0 |

**Select Statement**

Write a program to retrieve all the records present in the User table and display those records in the specified format using the SELECT select.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**  
  
  
Create a class **User** with the following private attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| id | Long |
| name | String |
| contactDetail | String |
| username | String |
| password | String |

Generate appropriate **Getters**and **Setters**  
Generate **default**and **parameterized constructors**  
The parameterized constructor should be in the following format. **User(Long id,String name, String contactDetail, String username, String password)**

Create a class **UserDAO** with the following methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| public List<User> getAllUsers() | This function finds all records of the User table and returns the User objects in a List. |

Create a driver class **Main** to display a List of objects in the following format.

NOTE : Use **System.out.format("%-5s %-5s %-15s %-10s %s\n","Id","Name","Contact Detail","Username","Password")** to display user details.  
  
**Table properties:**

CREATE  TABLE “user”(  
id number(19) NOT NULL,  
name VARCHAR2(45) NOT NULL,  
contact\_detail VARCHAR2(45) NOT NULL,  
username VARCHAR2(45) NOT NULL,  
password VARCHAR2(45) NOT NULL,  
PRIMARY KEY (id));  
CREATE SEQUENCE user\_seq START WITH 1 INCREMENT BY 1;

**Use the following code snippet to establish DBConnection:**

import java.util.ResourceBundle;

ResourceBundle rb = ResourceBundle.getBundle("oracle");

String url = rb.getString("db.url");

String username = rb.getString("db.username");

String password = rb.getString("db.password");

**oracle.properties:**

db.url = jdbc:oracle:thin:@localhost:1521:xe  
db.username = root  
db.password = student

**Download the oracle jar file in the below link.**  
[Oracle jar](http://hcl.e-box.co.in/uploads/ojdbc6.jar.zip)

**[All text in bold corresponds to the input and rest corresponds to the output]**

**Sample Input and Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Name | Contact Detail | Username | Password |
| 1 | John | 9876543210 | johny | 12345 |
| 2 | Peter | 9873216540 | peterey | pet123 |
| 3 | Adam | 9871236504 | adamanta | ad@123 |
| 4 | Linda | 8794561320 | lindahere | abcd |
| 5 | Tony | 7894561230 | tonii | abc123 |

**Using PreparedStatement**

Let’s try the PreparedStatement interface in this problem. The PreparedStatement interface enables you to perform Database operations by obtaining parameters at run-time. Let's have practice in PreparedStatement in the following exercise.

**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**

Create a class called **User** with the following private attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| id | Long |
| name | String |
| contactDetail | String |
| username | String |
| password | String |

Generate **getters**and **setters**.

Generate **Default**and two **parameterized constructors**.

Format for Parameterized constructors are

**User(String name, String contactDetail, String username, String password)**

**User(Long id, String name, String contactDetail, String username,String password)**

Create a class called **UserDAO** with the following methods

|  |  |
| --- | --- |
| **Method name** | **Description** |
| public void insertDetails(User u) | This method accepts a User object as an argument and inserts the details in the user table. |
| public List<User> getAllUser() | This method retrieves all user details in the ascending order of id from the user table, stores it in the list of user objects, and returns the list. |

Create a driver class called **Main**. In the main method, accept user details from the user and call appropriate methods in the **UserDAO**class.

**Table properties:**

CREATE  TABLE "user"(  
id number(19) NOT NULL,  
name VARCHAR2(45) NOT NULL,  
contact\_detail VARCHAR2(45) NOT NULL,  
username VARCHAR2(45) NOT NULL,  
password VARCHAR2(45) NOT NULL,  
PRIMARY KEY (id));  
CREATE SEQUENCE "user\_seq" START WITH 1 INCREMENT BY 1;

**Use the following code snippet to establish DBConnection:**  
import java.util.ResourceBundle;  
ResourceBundle rb = ResourceBundle.getBundle("oracle");  
String url = rb.getString("db.url");  
String username = rb.getString("db.username");  
String password = rb.getString("db.password");

**oracle.properties:**  
db.url = jdbc:oracle:thin:@localhost:1521:xe  
db.username = root  
db.password = student

**Download the oracle jar file in the below link.**  
[Oracle jar](http://hcl.e-box.co.in/uploads/ojdbc6.jar.zip)

**Note:   
  
while retrieving data from table sort by id.**  
Use **PreparedStatement** for queries.  
**System.out.format("%-5s %-10s %-15s %-10s %s\n","Id","Name","Contact Detail","Username","Password");**

**Sample Input and Output:  
[All text in bold corresponds to the input and rest corresponds to the output]**

Enter the user detail in CSV format

**Antony,9873216540,Antonie,an@987**

Id    Name       Contact Detail  Username   Password

1     John       9876543210      johny      12345

2     Peter      9873216540      peterey    pet123

3     Adam       9871236504      adamanta   ad@123

4     Linda      8794561320      lindahere  abcd

5     Tony       7894561230      tonii      abc123

6     Antony     9873216540      Antonie    an@987

**Insert with User identification**

In the previous problem, we stored data in a single table. But now we are gonna use 2 tables. One is the User and another is Hall. A User will be the owner of a Hall. So, the user object is kept in the hall class and the user id is kept in the hall table. We need to get the username of the owner, retrieve the object from the user table and get the details of the hall and place the user object in it as owner. Then store the hall details in the hall table with the user id. This will be helpful for the hall owners to register their halls for the event in our application. Let's get to programming this feature.  
  
Create a class **User** with the following private attributes,

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| id | Long |
| name | String |
| mobileNumber | String |
| username | String |
| password | String |

Add appropriate getter/setter, default and parameterized constructor.  
The parameterized constructor format **User(Long id, String name, String mobileNumber, String username, String password)**

Create a class **Hall** with the following private attributes,

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| id | Long |
| name | String |
| contactNumber | String |
| costPerDay | Double |
| owner | User |

Add appropriate getter/setter, default and parameterized constructor.  
The parameterized constructor format **Hall(String name, String contactNumber, Double costPerDay,User owner)**  
Override **toString**() and print the details in the tabular form.  
      
Create a class **UserDAO** with the following methods,

|  |  |
| --- | --- |
| **Method** | **Description** |
| User getUser(String username) | This method gets the user details from the user table whose username is passed as an argument. It creates a user object from those details and returns the object. If the user with that username not available, it returns null. |

Create a class **HallDAO** with the following methods,

|  |  |
| --- | --- |
| **Method** | **Description** |
| void saveHall(Hall hall) | This method gets the hall object as an argument and inserts it into the hall table. |
| List<Hall> getAllHall() | Returns the list of halls from database. |

Create a class **DbConnection** with the following static methods,

|  |  |
| --- | --- |
| **Method** | **Description** |
| static Connection getConnection() | This method uses "mysql.properties" file as a resource file and gets a connection object for the MySQL database and returns the connection object. |

Create a driver class **Main** and use the main method for user interactions.

**Table properties:**

CREATE  TABLE "user"(  
id number(19) NOT NULL,  
name VARCHAR2(45) NOT NULL,  
contact\_detail VARCHAR2(45) NOT NULL,  
username VARCHAR2(45) NOT NULL,  
password VARCHAR2(45) NOT NULL,  
PRIMARY KEY (id));  
CREATE SEQUENCE "user\_seq" START WITH 1 INCREMENT BY 1;

CREATE  TABLE hall(  
id number(19) NOT NULL,  
name VARCHAR2(255) NOT NULL,  
contact\_detail VARCHAR2(255) NOT NULL,  
cost\_per\_day BINARY\_DOUBLE NOT NULL,  
owner\_id NUMBER(19) NOT NULL,  
foreign key(owner\_id) references "user"(id),  
PRIMARY KEY (id));  
CREATE SEQUENCE hall\_seq START WITH 1 INCREMENT BY 1;

**Use the following code snippet to establish DBConnection:**  
import java.util.ResourceBundle;  
ResourceBundle rb = ResourceBundle.getBundle("oracle");  
String url = rb.getString("db.url");  
String username = rb.getString("db.username");  
String password = rb.getString("db.password");

**oracle.properties:**  
db.url = jdbc:oracle:thin:@localhost:1521:xe  
db.username = root  
db.password = student

**Download the oracle jar file in the below link.**  
[Oracle jar](http://hcl.e-box.co.in/uploads/ojdbc6.jar.zip)

**Input and Output format:**  
The first line is the hall details in the CSV format**(name, contact, cost per day).**  
The next line of input is the username.  
If the username doesn’t exist, then display **Username seems to be wrong!! Enter the correct username:**  
Otherwise, display the hall details of the corresponding owner.  
Refer to Sample Input and Output for further details and format of the output.  
Use "**%-15s%-15s%-15s%-15s**" while displaying hall objects in tabular form.  
  
**[All Texts in bold corresponds to the input and rest are output]  
Sample Input and Output 1:**  
  
Enter the details of hall in csv format:  
**Ball Room,1234567890,15000**  
Enter the username:  
**jim**  
Username seems to be wrong!! Enter the correct username:  
**johny**  
The hall details are:  
Name           Mobile         Cost           Owner            
Party hall     9874653201     5000.0         John             
Dining Hall    9876541230     3000.0         Peter            
Disco Hall     9871234560     8000.0         Adam             
Conference Hall7891236540     7500.0         Linda            
Meeting Hall   8974102365     9000.0         Tony             
Ball Room      1234567890     15000.0        John

**Update User Detail**

There was a small error in the user enrollment application. The mobile numbers of the users got interchanged unexpectedly. Hence create a console application to update the mobile number of the user by obtaining input from the console. If the username is present in the User table, get the contactDetail and update it. Otherwise, Display "No such user is present".  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**

Create a class **User** with the following private attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| id | Long |
| name | String |
| contactDetail | String |
| username | String |
| password | String |

Generate appropriate **Getters**and **Setters**

Generate **default** and **parameterized constructors**

Parameterized constructor should be in the following format

**User(Long id,String name, String contactDetail, String username, String password)**

Create a class **UserDAO** with the following methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| public List<User> getAllUsers() | This function finds all records of the User table and returns the User objects in a List. |
| public User findUserByUsername(String username) | This function accepts username as an argument and finds a record that matches the username and returns the User object. |
| public void updateUser(User user) | This function accepts the User object as an argument and updates the contactDetail for the record. |

Create a driver class **Main** to display a list of objects in the following format.

**Table Properties:**

CREATE  TABLE "user"(  
id number(19) NOT NULL,  
name VARCHAR2(45) NOT NULL,  
contact\_detail VARCHAR2(45) NOT NULL,  
username VARCHAR2(45) NOT NULL,  
password VARCHAR2(45) NOT NULL,  
PRIMARY KEY (id));  
CREATE SEQUENCE "user\_seq" START WITH 1 INCREMENT BY 1;

**Use the following code snippet to establish DBConnection:**

import java.util.ResourceBundle;

ResourceBundle rb = ResourceBundle.getBundle("oracle");

String url = rb.getString("db.url");

String username = rb.getString("db.username");

String password = rb.getString("db.password");

**oracle.properties:**

db.url = jdbc:oracle:thin:@localhost:1521:xe  
db.username = root  
db.password = student

**Download the oracle jar file in the below link.**  
[Oracle jar](http://hcl.e-box.co.in/uploads/ojdbc6.jar.zip)

**Input and Output format:**  
Display all the user details from the user table as the specified format.

Use System.out.format("**%-5s %-5s %-15s %-10s %s\n"**,"Id","Name","Contact Detail","Username","Password") to display user details.  
Then the input is the username of the user.  
Display the details of the corresponding user. If the username is not in the table, then display “**No such user is present**”  
Then the input is a mobile number to be updated for the particular user.

**[All text in bold corresponds to the input and rest corresponds to the output]**

**Sample Input and Output 1:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Name | Contact Detail | Username | Password |
| 1 | John | 9876543210 | johny | 12345 |
| 2 | Peter | 9873216540 | peterey | pet123 |
| 3 | Adam | 9871236504 | adamanta | ad@123 |
| 4 | Linda | 8794561320 | lindahere | abcd |
| 5 | Tony | 7894561230 | tonii | abc123 |

Enter the username:

**johny**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Name | Contact Detail | Username | Password |
| 1 | John | 9876543210 | johny | 12345 |

Enter the mobile number to be updated:

**9477885140**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Name | Contact Detail | Username | Password |
| 1 | John | 9477885140 | johny | 12345 |

**Sample Input and Output 2:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Id | Name | Contact Detail | Username | Password |
| 1 | John | 9876543210 | johny | 12345 |
| 2 | Peter | 9873216540 | peterey | pet123 |
| 3 | Adam | 9871236504 | adamanta | ad@123 |
| 4 | Linda | 8794561320 | lindahere | abcd |
| 5 | Tony | 7894561230 | tonii | abc123 |

Enter the username:

**admin**

No such user is present

**Delete based on condition**

Write a program to get the username and delete the user from the user table in the database.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**  
  
Create a class **User** with the following private attributes,

|  |  |
| --- | --- |
| **Attributes** | **Data type** |
| id | Long |
| name | String |
| mobileNumber | String |
| username | String |
| password | String |

Add appropriate getter/setter, default and parameterized constructor.  
Override **toString()** and print the details in the tabular form.  
  
  
Create a class **UserDAO** with the following methods,

|  |  |
| --- | --- |
| **Method** | **Description** |
| List<User> getAllUser() | this method gets the user details from the user table and create user objects using the details and add the user objects in a list and returns the list. |
| Boolean deleteUser(String username) | this method gets username as the argument and deletes the user with the given username from the table and returns true. If the user with the given username is not available it returns false. |

Create a class **DbConnection** with the following static methods,

|  |  |
| --- | --- |
| **Method** | **Description** |
| static Connection getConnection() | this method uses "oracle.properties" file as a resource file and gets connection object for the Oracle database and return the connection object. |

Create a driver class **Main** and use the main method for user interactions.

**Table properties:**

CREATE  TABLE "user"(  
id number(19) NOT NULL,  
name VARCHAR2(45) NOT NULL,  
contact\_detail VARCHAR2(45) NOT NULL,  
username VARCHAR2(45) NOT NULL,  
password VARCHAR2(45) NOT NULL,  
PRIMARY KEY (id));  
CREATE SEQUENCE "user\_seq" START WITH 1 INCREMENT BY 1;

The class **DbConnection** use the following resource bundle access,  
        ResourceBundle rb = ResourceBundle.getBundle("oracle");  
        String url = rb.getString("db.url");  
        String username = rb.getString("db.username");  
        String password = rb.getString("db.password");  
  
And the **oracle.properties** file has following data,  
       db.url = jdbc:oracle:thin:@localhost:1521:xe  
       db.username = root  
       db.password = student

**Download the oracle jar file in the below link.**  
[Oracle jar](http://hcl.e-box.co.in/uploads/ojdbc6.jar.zip)  
  
**Input and Output format:**  
Display the user details from the user table as the specified format.  
Use "**%-15s%-15s%-15s%-15s**" while displaying hall objects in tabular form.  
Then the input is the username of the user to be deleted from the user table.  
Print "**User not found**" if the method deleteUser returns false, else print "**User deleted successfully**" in main.  
Then display the user details after deletion from the user table.  
Refer to sample input and output for further details and format of the output.  
  
**[All Texts in bold corresponds to the input and rest are output]  
Sample Input and Output 1:**  
  
Name           Mobile         Username       Password         
John           9876543210     johny          12345            
Peter          9873216540     peterey        pet123           
Adam           9871236504     adamanta       ad@123           
Linda          8794561320     lindahere      abcd             
Tony           7894561230     tonii          abc123           
Enter the username to be deleted:  
**Jim**  
User not found  
Name           Mobile         Username       Password         
John           9876543210     johny          12345            
Peter          9873216540     peterey        pet123           
Adam           9871236504     adamanta       ad@123           
Linda          8794561320     lindahere      abcd             
Tony           7894561230     tonii          abc123           
  
**Sample Input and Output 2:**  
  
Name           Mobile         Username       Password         
John           9876543210     johny          12345            
Peter          9873216540     peterey        pet123           
Adam           9871236504     adamanta       ad@123           
Linda          8794561320     lindahere      abcd             
Tony           7894561230     tonii          abc123           
Enter the username to be deleted:  
**lindahere**  
User deleted successfully  
Name           Mobile         Username       Password         
John           9876543210     johny          12345            
Peter          9873216540     peterey        pet123           
Adam           9871236504     adamanta       ad@123           
Tony           7894561230     tonii          abc123

**List of Stalls**

Now we are gonna get the details of a table with the id of another table. There are many exhibitions in our application and each exhibition has booked many stalls. So we want a consolidated record of it from our table. So write a program to get all the stalls booked for the given exhibition.  
  
Create a class **Exhibition** with the following private attributes,

|  |  |
| --- | --- |
| **Attributes** | **Data type** |
| id | Long |
| name | String |
| stallList | List<Stall> |

Add appropriate getter/setter, default and parameterized constructor.

Create a class **Stall** with the following private attributes,

|  |  |
| --- | --- |
| **Attributes** | **Data type** |
| id | Long |
| name | String |
| detail | String |
| owner | String |
| exhibition | Exhibition |

Add appropriate getter/setter, default and parameterized constructor.  
  
Create a class **ExhibitionDAO** with following methods,

|  |  |
| --- | --- |
| **Method** | **Description** |
| Exhibition getExhibition(String name) | This method takes exhibition name as an argument, retrieves the exhibition detail from the exhibition table and get all the stalls corresponding to the exhibition and stores it in the exhibition object and returns the object. |

Create a class **DbConnection** with the following static methods,

|  |  |
| --- | --- |
| **Method** | **Description** |
| static Connection getConnection() | this method uses "oracle.properties" file as a resource file and gets connection object for the MySQL database and returns the connection object. |

Create a driver class **Main** and use the main method for the user interactions.

**Table properties:**

CREATE  TABLE exhibition(  
id NUMBER(19) NOT NULL,  
name VARCHAR2(45) NOT NULL,  
PRIMARY KEY(id));  
CREATE SEQUENCE exhibition\_seq START WITH 1 INCREMENT BY 1;

CREATE TABLE stall(  
id NUMBER(19) NOT NULL,  
name VARCHAR2(255) NOT NULL,  
detail VARCHAR2(255) NOT NULL,  
owner VARCHAR2(255) NOT NULL,  
exhibition\_id NUMBER(19) NOT NULL,  
foreign key(exhibition\_id) references exhibition(id),  
PRIMARY KEY (id));  
CREATE SEQUENCE stall\_seq START WITH 1 INCREMENT BY 1;

The class **DbConnection** use the following resource bundle access,  
        ResourceBundle rb = ResourceBundle.getBundle("oracle");  
        String url = rb.getString("db.url");  
        String username = rb.getString("db.username");  
        String password = rb.getString("db.password");  
  
And the **oracle.properties** file has following data,  
       db.url = jdbc:oracle:thin:@localhost:1521:xe  
       db.username = root  
       db.password = student

**Download the oracle jar file in the below link.**  
[Oracle jar](http://hcl.e-box.co.in/uploads/ojdbc6.jar.zip)  
  
**Input and Output format:**  
The input is the exhibition name.  
If the entered exhibition name is not in the table, then display “**Enter the correct exhibition name**:”  
Otherwise, display the stall details corresponding to the given exhibition.  
Use "**%-20s%-20s%-15s**" as formatting to display the stall list in tabular format.  
 Refer to sample input/output for further details and format of the output.  
  
**[All Texts in bold corresponds to the input and rest are output]  
Sample Input and Output 1:**  
  
Enter the exhibition name:  
**Exhibition 1**  
Stall Name          Detail              Owner name       
Chocolate stall     chocolate shop      John             
HiFi electronics    mobile shop         Adam             
Snack seeker        shop for snacks     Mark

**Display Items by Category**

The Sample data for the Items and ItemTypes are updated by the organizers. There is a need for a feature that would fetch the Items that belong to a particular ItemType from the database and display it to the users. Thus compiled a list of items would be used for advertisements of the exhibitions. Hence write a console application to display the specified information.  
  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**

Create a class **ItemType** with the following private attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| id | Long |
| name | String |
| deposit | Double |
| costPerDay | Double |

Create a class **Item** with the following private member attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| id | Long |
| name | String |
| itemType | ItemType |
| vendor | String |

Generate appropriate **Getters**and **Setters**for the above classes

Generate **default**and **parameterized constructors**

Parameterized constructor should be in the following format

**ItemType(Long id, String name, Double deposit, Double costPerDay)**

**Item(Long id, String name, ItemType itemType, String vendor)**

Create a class **ItemTypeDAO** with the following methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| public List<ItemType> getAllItemTypes() | This function finds all records of the ItemType table and returns the ItemType objects in a List. |

Create a class **ItemDAO** with the following methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| public List<Item> findItemsByCategory(String category) | This function accepts category as an argument and finds the records that match the given category and returns the Item objects in a List. |

Create a driver class **Main** to display a List of objects in the following format.  
  
**Table Structure:**

CREATE TABLE item\_type(  
id number(19) NOT NULL,  
name varchar2(255) NOT NULL,  
deposit BINARY\_DOUBLE NOT NULL,  
cost\_per\_day BINARY\_DOUBLE NOT NULL,  
PRIMARY KEY (id));  
CREATE SEQUENCE item\_type\_seq START WITH 1 INCREMENT BY 1;

CREATE  TABLE item(  
id number(19) NOT NULL,  
name VARCHAR2(255) NOT NULL,  
vendor VARCHAR2(255) NOT NULL,  
type\_id number(19) NULL,  
foreign key(type\_id) references item\_type(id),  
PRIMARY KEY (id));  
CREATE SEQUENCE item\_seq START WITH 1 INCREMENT BY 1;

**Use the following code snippet to establish DBConnection:**

import java.util.ResourceBundle;

ResourceBundle rb = ResourceBundle.getBundle("oracle");

String url = rb.getString("db.url");

String username = rb.getString("db.username");

String password = rb.getString("db.password");

**oracle.properties:**

db.url = jdbc:oracle:thin:@localhost:1521:xe  
db.username = root  
db.password = student

**Download the oracle jar file in the below link.**  
[Oracle jar](http://hcl.e-box.co.in/uploads/ojdbc6.jar.zip)

**Output format:**

Use **System.out.format("%-5s %-15s %-12s %s\n","ID","Name","Deposit","Cost per day")** to display ItemType details.  
Use **System.out.format("%-5s %-15s %-12s %s\n","ID","Name","Item Type","Vendor")** to display Item details.

**[All text in bold corresponds to the input and rest corresponds to the output]**

**Sample Input and Output 1:**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Deposit | Cost per day |
| 1 | Food | 50000.0 | 10000.0 |
| 2 | Electronics | 85000.0 | 15000.0 |
| 3 | Fashion | 36000.0 | 8000.0 |
| 4 | Grooming | 15000.0 | 5000.0 |
| 5 | Books | 20000.0 | 7500.0 |

Enter the category:

**Food**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Item Type | vendor |
| 1 | Chocolate | Food | Foodies Court |
| 2 | Lollypop | Food | Foodies Court |

**Sample Input and Output 2:**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Deposit | Cost per day |
| 1 | Food | 50000.0 | 10000.0 |
| 2 | Electronics | 85000.0 | 15000.0 |
| 3 | Fashion | 36000.0 | 8000.0 |
| 4 | Grooming | 15000.0 | 5000.0 |
| 5 | Books | 20000.0 | 7500.0 |

Enter the category:

**Mobiles**

No such category is present

Top of Form

Bottom of Form

**Update timings**

The event organizers had requested a feature for updating the Event details if any changes occur in the feature. Hence as a prototype, write a program for a console application that could prompt for the event timings and updates the respective timing changes in the database.   

**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**

Create a class called **Event** with the following private attributes.

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| id | Long |
| name | String |
| detail | String |
| startDate | java.util.date |
| endDate | java.util.date |
| organizer | String |

Generate getters and setters.

Generate Default and two parameterized constructors.

Format for Parameterized constructors are

**Event(String name, String detail, Date startDate, Date endDate,String organizer)**

**Event(Long id, String name, String detail, Date startDate,Date endDate, String organizer)**

Create a class called **EventDAO** with following methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| public Event getEventById(Long id) | This method accepts an id of Event as the argument and fetches the Event detail with the corresponding id. Then returns the Event details in an Event object. |
| public void updateEvent(Event e) | This method accepts the Event object as an argument and updates the Event details present in the object to the row that has the Event id as that of the id present in the object |
| public List<Event> getAllEvents() | This method retrieves all the Event details from the event table, stores the details in list of Event objects, and returns the list. |

Create a driver class called Main. In the main method, accept event details from the user and call appropriate methods in EventDAO class. When an id not present in the table is given print "Id not found " and terminate.  
  
  
**Table properties:**  
CREATE  TABLE event(  
id NUMBER(10) NOT NULL,  
name VARCHAR2(255) NOT NULL,  
detail VARCHAR2(255) NOT NULL,  
start\_date TIMESTAMP(0) NOT NULL,  
end\_date TIMESTAMP(0) NOT NULL,  
organizer VARCHAR2(255) NOT NULL,  
PRIMARY KEY (id));  
CREATE SEQUENCE event\_seq START WITH 1 INCREMENT BY 1;

**Use the following code snippet to establish DBConnection:**  
import java.util.ResourceBundle;  
ResourceBundle rb = ResourceBundle.getBundle("oracle");  
String url = rb.getString("db.url");  
String username = rb.getString("db.username");  
String password = rb.getString("db.password");  
  
**oracle.properties:**  
db.url = jdbc:oracle:thin:@localhost:1521:xe  
db.username = root  
db.password = student

**Download the oracle jar file in the below link.**  
[Oracle jar](http://hcl.e-box.co.in/uploads/ojdbc6.jar.zip)

**Input and Output format:**  
While retrieving data from table sort by id.  
System.out.format (**"%-5s %-10s %-15s %-20s %-20s %s\n"**,"Id","Event name","Detail","Start date","End date","Organizer");  
Then the input is event id to update the particular event details. If the event id is not present in the table then display “**Id not found**” and terminate the process.  
Then the input is startdate and end date of the events to be updated.  
Then display all the event details after updation.

**[All text in bold corresponds to the input and rest corresponds to the output]**

**Sample Input/Output:**

Id    Event name Detail          Start date           End date             Organizer

1     Event 1    event 1         2018-01-01 12:00:00  2018-02-01 12:00:00  John

2     Event 2    event 2         2018-02-15 18:00:00  2018-03-20 15:00:00  Peter

3     Event 3    event 3         2018-03-01 15:00:00  2018-04-01 00:00:00  Mark

4     Event 4    event 4         2018-03-01 12:00:00  2018-03-10 18:00:00  Stephen

5     Event 5    event 5         2018-11-11 00:00:00  2018-12-01 12:00:00  Charles

Enter the id of the event to be updated

**1**

Enter the start and end date

**2018-01-02 12:00:00**

**2018-01-03 12:00:00**

Id    Event name Detail          Start date           End date             Organizer

1     Event 1    event 1         2018-01-02 12:00:00  2018-01-03 12:00:00  John

2     Event 2    event 2         2018-02-15 18:00:00  2018-03-20 15:00:00  Peter

3     Event 3    event 3         2018-03-01 15:00:00  2018-04-01 00:00:00  Mark

4     Event 4    event 4         2018-03-01 12:00:00  2018-03-10 18:00:00  Stephen

5     Event 5    event 5         2018-11-11 00:00:00  2018-12-01 12:00:00  Charles

**Hall insert**

When the number of rows to insert into the table increases, we can opt for multiple insert using batch queries. Hence instead of adding single row at a time, we can insert multiple rows at the same time which increases performance considerably. To get acquainted in batch query, obtain multiple input from the user and use Batch query for inserting mulitple hall details in this exercise.

Create a class called **Hall** with the following private attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| id | Long |
| name | String |
| contactDetail | String |
| costPerDay | Double |
| owner | String |

Generate **getters**and **setters**.

Generate **Default**and two **parameterized constructors**.

Format for Parameterized constructors are

**Hall(String name, String contactDetail, Double costPerDay, String owner)**

**Hall(Long id, String name, String contactDetail, Double costPerDay,String owner)**

Create a class called **HallDAO** with the following methods

|  |  |
| --- | --- |
| **Method name** | **Description** |
| public List<Hall> getHallList() | This method retrieves all the Hall details from the hall table, stores the details in list of Hall objects, and returns the list. |
| public void bulkInsert(List<Hall> list) | This method accepts list of Hall objects as argument and stores each object details into the hall table. |

Create a driver class called Main. In the main method, accept event details from the user and call appropriate methods in HallDAO class.

**Note:** while retrieving data from table sort by cost\_per\_day.

Use batch for inserting multiple rows.  
**System.out.format ("%-5s %-15s %-15s %-15s %s\n","Id","Hall name","Contact detail","Cost per day","Organizer");**

**Table properties:**

CREATE  TABLE hall(  
id number(19) NOT NULL,  
name VARCHAR2(255) NOT NULL,  
contact\_detail VARCHAR2(255) NOT NULL,  
cost\_per\_day BINARY\_DOUBLE NOT NULL,  
owner VARCHAR2(255) NOT NULL,  
PRIMARY KEY (id));  
CREATE SEQUENCE hall\_seq START WITH 1 INCREMENT BY 1;

**Use the following code snippet to establish DBConnection:**  
import java.util.ResourceBundle;  
ResourceBundle rb = ResourceBundle.getBundle("oracle");  
String url = rb.getString("db.url");  
String username = rb.getString("db.username");  
String password = rb.getString("db.password");  
  
**oracle.properties:**  
db.url = jdbc:oracle:thin:@localhost:1521:xe  
db.username = root  
db.password = student

**Download the oracle jar file in the below link.**  
[Oracle jar](http://hcl.e-box.co.in/uploads/ojdbc6.jar.zip)

**[Strictly adhere to the Object-Oriented Specifications given in the problem statement.**

**All class names, attribute names and method names should be the same as specified in the problem statement.]**

**[All text in bold corresponds to the input and rest corresponds to the output]**

**Sample Input/Output:**

Enter the number of hall details:

**2**

Enter hall 1 detail in CSV format

**HH Hall,9985471320,15000,Ram**

Enter hall 2 detail in CSV format

**RR Hall,9876543210,1000,Mahesh**

Id    Hall name       Contact detail  Cost per day    Organizer

7     RR Hall         9876543210      1000.0          Mahesh

2     Dining Hall     9876541230      3000.0          Peter

1     Party hall      9874653201      5000.0          John

4     Conference Hall 7891236540      7500.0          Linda

3     Disco Hall      9871234560      8000.0          Adam

5     Meeting Hall    8974102365      9000.0          Tony

6     HH Hall         9985471320      15000.0         Ram

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form