**---------------------------------TYPE 1-------------------------------------------**

**Alphabet Game 3**

Rina is a very innovative KG teacher and she loves game based teaching. She has taught her KG kids lower case alphabets.

She wanted to revise lower case alphabets. For that she has devised a new game called Lucky Alphabet Game.

The Lucky Alphabet Game is played as follows: At the beginning, several kids enter the field. Each kid has a lowercase English letter pasted on their back. The game is played in turns. In each turn, Rina says a letter such that there are atleast 'n' kids (with that letter pasted on their back in the field) and 'n' kids with that letter pasted on the back leave the field. The game ends once it is impossible to take another turn.

If there are some kids left in the field at the end of the game, they must all have the different letters. Such letters are called lucky alphabets. If there are no kids left in the field at the end of the game, there is no lucky alphabet.

Write a program to find the lucky alphabets.

**Assume**: Maximum number of kids <= 150.

**Input and Output Format:**

The first line of the input consists of a string that contains the letters pasted on the back of the kids.

The second line of the input consists of an integer that corresponds to n.

Output consists of the lucky alphabets sorted in ascending order.

If there are no lucky alphabets, print 'NO'.

**Sample Input:**

asdad

2

**Sample Output:**

s

**Alphabet Game 4**

Rina is a very innovative KG teacher and she loves game based teaching. She has taught her KG kids lower case alphabets.

She wanted to revise lower case alphabets. For that she has devised a new game called Lucky Alphabet Game.

The Lucky Alphabet Game is played as follows: At the beginning, several kids enter the field. Each kid has a lowercase English letter pasted on their back. The game is played in turns. In each turn, Rina says a letter such that there are 2 kids , one kid with that letter and the other kid with one of its adjacent letters and the 2 kids leave the field. The game ends once it is impossible to take another turn. Rina is quite intelligent and she always selects letters such that only minimal number of kids are left in the field at the end of the game.

If there are some kids left in the field at the end of the game, they must all have the nonadjacent letters. Such letters are called lucky alphabets. If there are no kids left in the field at the end of the game, there is no lucky alphabet. If duplicates are present, print once.

Write a program to find the lucky alphabets.

**Assume**: Maximum number of kids <= 150.

**Input and Output Format:**

Input consists of a string that contains the letters pasted on the back of the kids.

Output consists of the lucky alphabets sorted in ascending order.

If there are no lucky alphabets, print 'NO'.

**Sample Input:**

abdcd

**Sample Output:**

d

**Alphabet Game 1**

Rina is a very innovative KG teacher and she loves game based teaching. She has taught her KG kids lower case alphabets.

She wanted to revise lower case alphabets. For that she has devised a new game called Lucky Alphabet Game.

The Lucky Alphabet Game is played as follows: At the beginning, several kids enter the field. Each kid has a lowercase English letter pasted on their back. The game is played in turns. In each turn, Rina says one of the letters of the kids present in the field and all kids with letters that appear before this letter (the letter that Rina said) in lexicographic order leave the field. The game ends when there is only one kid left in the field. That kid is the winner and the letter that the kid has is the Lucky Alphabet.

Write a program to find the lucky alphabet.

**Input and Output Format:**

The first line of the input consists of a string that contains the letters pasted on the back of the kids.

The other inputs correspond to the letter that Rina selects in every turn.

After each input selection, list the letters of the kids in the field.

The last line corresponds to the lucky alphabet.

**Note** : If the input string contains only one character, don't take any further inputs. Just print the string.

**Assumption** : Assume that all letters in the input string are unique.

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

**Sample Input and Output:**

**abcfghde**

**b**

bcfghde

**f**

fgh

**h**

h

h

**Alphabet Game 2**

Rina is a very innovative KG teacher and she loves game based teaching. She has taught her KG kids lower case alphabets.

She wanted to revise lower case alphabets. For that she has devised a new game called Lucky Alphabet Game.

The Lucky Alphabet Game is played as follows: At the beginning, several kids enter the field. Each kid has a lowercase English letter pasted on their back. The game is played in turns. In each turn, Rina says one of the letters of the kids present in the field and all kids with letters that appear after this letter (the letter that Rina said) in lexicographic order leave the field. The game ends when there is only one kid left in the field. That kid is the winner and the letter that the kid has is the Lucky Alphabet.

Write a program to find the lucky alphabet.

**Input and Output Format:**

The first line of the input consists of a string that contains the letters pasted on the back of the kids.

The other inputs correspond to the letter that Rina selects in every turn.

After each input selection, list the letters of the kids in the field.

The last line corresponds to the lucky alphabet.

Assumption : Assume that all letters in the input string are unique.

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

**Sample Input and Output:**

**abcfghde**

**b**

ab

**a**

a

a

**SMS Language**

SMS language or textese (also known as txt-speak, txtese, chatspeak, txt, txtspk, txtk, txto, textinglanguage, txt lingo, SMSish, txtslang,or txt talk) is a term for the abbreviations and slang commonlyused with mobile phone text messaging.

Some of the abbreviations used are

* s for yes
* u for you
* 2day for today
* y for why

Many grandpa's have started sending SMSes to their grand children. But they are not familiar withthe SMS lingo.

Can you help them by writing a program that would convert a given text in proper English to SMSlingo? Consider only the 4 words listed above.

**Input Format:**

Input consists of a single string. Assume that the maximum length of the string is 200 and all lettersare in lower-case.

**Output Format:**

Output consists of a single string.

**Sample Input 1:**

where were you yesterday?

**Sample Output 1:**

where were u sterday?

**Sample Input 2:**

why is today a working day for you?

**Sample Output 2:**

y is 2day a working day for u?

Hint:

In C programming, use %[^\n]s as the format specifier for reading strings with spaces.  [%s is used to read strings without any spaces in between]    or use gets() function.

**---------------------------------TYPE 2-------------------------------------------**

Comparable - Purchased Items

Write a Java program **to list all the items purchased by the customer based on price (High to Low)**.  Comparable interface is used to defined the natural ordering of objects. Implement Comparable interface and implement compareTo() method which returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.  
  
Get all the purchased items and their corresponding price from the user and store it in a list. Use Collections.sort() method to sort the list and display the result as shown in the sample input and output.  
  
Create a main class "Main.java"  
Create Item class with the attributes name(String) and price(Double) implementing Comparable interface and implementing compareTo() method  
  
Include a constructor with the arguments item name and price  
Include appropiate getter and setter for the Item class  
  
**Input and Output Format:**  
First input corresponds to the number of items purchased by the customer.  
Display the item details separated by single space.  
Refer sample input and output for formatting specifications.  
  
**[All text in bold corresponds to input and the rest corresponds to output]**  
**Sample Input/Output :**  
Please provide the number of items purchased  
**3**  
Enter the name of the item 1  
**Rice**  
Enter the price of the item 1  
**50**  
Enter the name of the item 2  
**Dhal**  
Enter the price of the item 2  
**78**  
Enter the name of the item 3  
**Egg**  
Enter the price of the item 3  
**100**  
Item Details by price(High to Low)  
1 Egg 100.0  
2 Dhal 78.0  
3 Rice 50.0

Comparable - List Complaint By Date

Write a Java program to get all the complaint from the users and display the complaint sorted based on date (desc). The complaint consist of customer name,  complaint text, date of complaint and severity of the complaint. Display a menu of severity (Low, Medium and High) and user choose any one of the option.  
  
Create a main class "Main.java"  
Create Complaint class with below members,

* customer - String
* complaintText - String
* dateOfComplaint - Date
* severity - Integer

Add appropriate getter and setter methods for Complaint class  
*Example:   
Attribute - name  
Method - getName(), setName(String name)*  
Include  a constructor for Complaint with the arguments customer, complaint text, date of complaint and severity  
Implement Comparable interface and implement compareTo method to do the sorting based on the date  
  
**Input and Output Format:**  
First input corresponds to the number of contacts and followed by each contact information.  
Refer sample input and output for formatting specifications.  
  
**[All text in bold corresponds to input and the rest corresponds to output]**  
**Sample Input/Output :**  
Please provide the number of complaints to be registered  
**5**  
Please provide your name  
**Pradeep**  
Type in your complaint  
**The product is shipped with damage**  
Please select the severity of your complaint  
1.Low  
2.Medium  
3.High  
**3**  
Please provide Date in (MM-dd-yyyy)  
**05-12-2016**  
Please provide your name  
**Navin**  
Type in your complaint  
**Delay in delivery of the product**  
Please select the severity of your complaint  
1.Low  
2.Medium  
3.High  
**2**  
Please provide Date in (MM-dd-yyyy)  
**05-11-2016**  
Please provide your name  
**Kavin**  
Type in your complaint  
**Damaged product**  
Please select the severity of your complaint  
1.Low  
2.Medium  
3.High  
**2**  
Please provide Date in (MM-dd-yyyy)  
**05-23-2016**  
Please provide your name  
**Sudha**  
Type in your complaint  
**Product is not yet delivered**  
Please select the severity of your complaint  
1.Low  
2.Medium  
3.High  
**3**  
Please provide Date in (MM-dd-yyyy)  
**06-02-2016**  
Please provide your name  
**Karthi**  
Type in your complaint  
**Delay in the product delivery**  
Please select the severity of your complaint  
1.Low  
2.Medium  
3.High  
**1**  
Please provide Date in (MM-dd-yyyy)  
**05-20-2016**  
Complaint Details  
Name Sudha  
Complaint raised on 06-02-2016  
Product is not yet delivered  
Name Kavin  
Complaint raised on 05-23-2016  
Damaged product  
Name Karthi  
Complaint raised on 05-20-2016  
Delay in the product delivery  
Name Pradeep  
Complaint raised on 05-12-2016  
The product is shipped with damage  
Name Navin  
Complaint raised on 05-11-2016  
Delay in delivery of the product  
  
  
  
**-------------**

Comparable - Display State

Write a Java program to get the country names and state names from the user seperated by a pipe symbol. Finally display all the countries and their states sorted in ascending order based on their names.  
  
Create a main class "Main.java"  
Create country class with below attributes,

* name - String
* stateList - List<State> (All state object for this country is stored in this list)

Add appropriate getter and setter methods for **Country** class  
Include a constructor accepting country name as a parameter  
Below are the methods in country class

* addState(String statename) - Add the new state to this country object
* getStateList() - Sort the state collection and return the list

Create **State** class with single attribute **name**  
Add appropriate getter and setter methods for State class  
Include a constructor with single argument state name  
Implement Comparable interface in the State class and implement the method compareTo()  
  
**Input and Output Format:**  
First input corresponds to the number of input elements and followed by country and state information in the format countryname|statename.  
Display the state name followed by two hyphen(-)  
Refer sample input and output for formatting specifications.  
  
**[All text in bold corresponds to input and the rest corresponds to output]**  
**Sample Input/Output :  
10  
India|Tamilnadu  
India|Kerala  
India|Odisha  
USA|Texas  
USA|Mississippi  
USA|Alaska  
India|Punjab  
Australia|Victoria  
Australia|Tasmania  
Australia|Queensland**  
Countries and States in ascending order  
Australia  
--Queensland  
--Tasmania  
--Victoria  
India  
--Kerala  
--Odisha  
--Punjab  
--Tamilnadu  
USA  
--Alaska  
--Mississippi  
--Texas  
**------------------------**

 Comparable - Contact Information Based on Mobile Number

Write a Java program to read all the contact information from the user and display the contact name and their mobile number sorted based on their mobile number (descending order). The contact details consist of name, email, mobile and address. Use Collections.sort() method for sorting.

Create a main class "Main.java"  
Create Contact class with below attributes  
name - String  
email - String  
mobile - Long  
address - String  
Add appropriate getter and setter methods for Contact class  
Include a constructor for Contact class with the arguments name, email, mobile and address  
Implment Comparable interface and implement the method compareTo() to perform sorting based on mobile number

Input and Output Format:  
First input corresponds to the number of contacts and followed by each contact information.  
Refer sample input and output for formatting specifications.

[All text in bold corresponds to input and the rest corresponds to output]  
Sample Input/Output :  
Enter number of contacts:  
2  
Enter contact 1 detail  
Enter Name  
Amar  
Enter Email  
[amar@gmail.com](https://mail.cognizant.com/owa/redir.aspx?REF=NrLMBWWMmGsQZAHZuxVxQ7lAECsTcwEMWskF1qMMzreYcME-PAbUCAFtYWlsdG86YW1hckBnbWFpbC5jb20.)  
Enter Mobile  
7200762700  
Enter Address  
Coimbatore  
Enter contact 2 detail  
Enter Name  
Thana  
Enter Email  
[thana@gmail.com](https://mail.cognizant.com/owa/redir.aspx?REF=8HVC8uTzE53c4S6TDvtTHpTH4LCMmiYDSzg4hZPlHNOYcME-PAbUCAFtYWlsdG86dGhhbmFAZ21haWwuY29t)  
Enter Mobile  
9566905846  
Enter Address  
Karur  
Contact list after sort by mobile number in descending order  
Thana-9566905846  
Amar-7200762700

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

Problem Description

Comparator - Complaint List Based on Severity and Date of Complaint

Write a Java program to get all complaint details from the user and display a report based on the severity and date of complaint.  Higher severity complaint and older complaints should be display on the top.  
Display a menu to select the severity of the complaint as shown in the sample input and output.  
  
Create a main class "Main.java"  
Create **Complaint** class with below attributes,

* customer - String
* complaintText - String
* dateOfComplaint - Date
* severity - Integer

Add appropriate getter and setter methods for Complaint class  
Create constructor for Complaint class with arguments customer, complaint text, date of complaint and severity  
  
Create **ComplaintSeverityDateComparator** class implementing **Comparator** and implementing the below method,  
*public int compare(Complaint complaint1, Complaint complaint2);*  
The compare method compares the two complaint object based on the severity and date of complaint  
this method retruns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.  
  
Read all the inputs in the Main class and store the list of complaints in ArrayList. Use Collections.sort() method to sort the list and pass the custom comparator.  
  
  
**Input and Output Format:**  
First input corresponds to the number of complaints and followed by the details of the complaint.  
Refer sample input and output for formatting specifications.  
  
**[All text in bold corresponds to input and the rest corresponds to output]  
Sample Input/Output :**  
Please provide the number of complaints to be registered  
**4**  
Please enter customer name  
**Madhan**  
Complaint Details  
**Network connection is not proper**  
Please select the severity of complaint  
1.Low  
2.Medium  
3.High  
**3**  
Please provide Date in (dd-MM-yyyy)  
**05-07-2016**  
Please enter customer name  
**Amar**  
Complaint Details  
**Printer is not working**  
Please select the severity of complaint  
1.Low  
2.Medium  
3.High  
**3**  
Please provide Date in (dd-MM-yyyy)  
**24-06-2016**  
Please enter customer name  
**Amar**  
Complaint Details  
**Images is not properly displaying in Mails**  
Please select the severity of complaint  
1.Low  
2.Medium  
3.High  
**2**  
Please provide Date in (dd-MM-yyyy)  
**07-07-2016**  
Please enter customer name  
**Jimesh**  
Complaint Details  
**Lift out of service**  
Please select the severity of complaint  
1.Low  
2.Medium  
3.High  
**3**  
Please provide Date in (dd-MM-yyyy)  
**07-07-2016**  
Complaint Details  
Customer : Amar Date :24-06-2016 Severity : High  
Printer is not working  
Customer : Madhan Date :05-07-2016 Severity : High  
Network connection is not proper  
Customer : Jimesh Date :07-07-2016 Severity : High  
Lift out of service  
Customer : Amar Date :07-07-2016 Severity : Medium  
Images is not properly displaying in Mails

**---------------------------------TYPE 3-------------------------------------------**

#### Problem Description

TreeMap - Letter Frequency

Write a Java program to calculate the character frequency in a sentence. The input consist of a single sentence and the output display a graphical chart displaying the freqency of each character by number of asterisk (\*). Display the character in the output in alphabetical order. Compute the frequence of all letters except space.  
  
Use TreeMap to store the characters and frequency since the tree map maintains the entries sorted based on their natural ordering.  
  
Create a main class "Main.java"  
Create a class **LetterSequence** and include below methods and attributes,  
Include a constructor to get the sentence as the input

|  |  |
| --- | --- |
| **Method/Attribute** | **Details** |
| public TreeMap<Character,Integer> computeFrequency() | Compute the frequency of each character in the sentence and store it in the TreeMap. Return the TreeMap after the computation. |
| public void displayLetterFrequency(TreeMap<Character,Integer> frequencyMap) | Iterate the tree map and get all the entries and print the information in a graphical view as shown the sample output |
| private String sentence | Input sentence is stored in this attribute |

**Input and Output Format:**  
Refer sample input and output for formatting specifications.  
  
**[All text in bold corresponds to input and the rest corresponds to output]**  
**Sample Input/Output :**  
Enter the input string  
**Refer sample input and output for formatting specifications**  
R : \*  
a : \*\*\*\*  
c : \*\*  
d : \*  
e : \*\*\*\*  
f : \*\*\*\*  
g : \*  
i : \*\*\*\*\*  
l : \*  
m : \*\*  
n : \*\*\*\*  
o : \*\*\*\*  
p : \*\*\*\*  
r : \*\*\*  
s : \*\*\*  
t : \*\*\*\*\*\*  
u : \*\*\*

HashMap - Complaint Register

Write a Java program to register all the complaints from multiple users and provide a quick view of number of complaint based on the name of the user. Use HashMap to store all the complaints, Key will be the name of the user and value contains the user object. Multiple complaints for a user is provided using a delimiter pipe "|".  
  
Create a main class "Main.java"  
Create User class with below attributes,

* name - String
* complaintCount - Integer

Add appropriate getter and setter methods for User class  
Create constructor for User class with arguments name and complaint count  
  
Provide search option to search based on the name of the user and display the number of complaints registered for him.  
  
**Input and Output Format:**  
Refer sample input and output for formatting specifications.  
  
**[All text in bold corresponds to input and the rest corresponds to output]**  
**Sample Input/Output :**  
Enter the user name  
**Jimesh**  
Enter Complaints - seperated by "|" symbol.  
**Power problem|water leakage|industrial pollution**  
Do you want to add another user (yes/no)  
**yes**  
Enter the user name  
**Madhan**  
Enter Complaints - seperated by "|" symbol.  
**traffic problem**  
Do you want to add another user (yes/no)  
**no**  
Enter the user name to search  
**john**  
No user found with the name john  
Do you want to search another user (yes/no)  
**yes**  
Enter the user name to search  
**Jimesh**  
User name : Jimesh  
Complaint Count : 3  
Do you want to search another user (yes/no)  
**no**

HashMap - Complaint Register View

Write a Java program to register all the complaints from multiple users and provide a quick view of all the complaints based on the name of the user. Use HashMap to store all the complaints, Key will be the user object and value contains list of complaint object. Multiple complaints for a user is provided using a delimiter pipe "|".  
  
Create a main class "Main.java"  
Create **User** class with below attributes,

* name - String

Add appropriate getter and setter methods for User class  
Create constructor for User class with argument name  
  
Create **Complaint** class with below attributes,  
description - String  
user - User object  
Add appropriate getter and setter methods for Complaint class  
Create constructor for Complaint class with arguments description and user object  
  
Provide search option to search based on the name of the user and display all the complaints registered for him.  
  
**Input and Output Format:**  
Refer sample input and output for formatting specifications.  
  
**[All text in bold corresponds to input and the rest corresponds to output]**  
**Sample Input/Output :**  
Enter the user name  
**Jimesh**  
Enter Complaints - seperated by "|" symbol.  
**traffic near the school|industry pollution in village**  
Do you want to add another user (yes/no)  
**no**  
Enter the user name to search  
**John**  
No user found with the name John  
Do you want to search another user (yes/no)  
**yes**  
Enter the user name to search  
**Jimesh**  
Complaints of user Jimesh :  
1) traffic near the school  
2) industry pollution in village  
Do you want to search another user (yes/no)  
**no**

HashMap - Currency Conversion

Write a Java program for perform currency conversion, The input consist of integer INR amount and provide below conversion option to the user,  
1. US Dollar  
2. Euro  
3. Yen  
4. Exit  
  
Create a main class "Main.java"  
Get the INR input from the user and based on the selection convert the amount to the corresponding currency. Maintain the exchange rates of the currency in a HashMap as shown below,

|  |  |
| --- | --- |
| **Currency** | **Exchange Rate** |
| US Dollar | 0.0149 |
| Euro | 0.0135 |
| Yen | 1.5211 |

Show the conversion option menu until user provides the exit option. If the user enter any other value prompt the message "Invalid option" and again continue to show the conversion option.  
  
**Note:**  
Display the dollar value in two decimal places.  
**Input and Output Format:**  
Refer sample input and output for formatting specifications.  
  
**[All text in bold corresponds to input and the rest corresponds to output]**  
**Sample Input/Output :**  
Enter INR  
**2000**  
1.US Dollar  
2.Euro  
3.Yen  
4.Exit  
Enter Choice  
**1**  
Dollar value: 29.80  
1.US Dollar  
2.Euro  
3.Yen  
4.Exit  
Enter Choice  
**2**  
Dollar value: 27.00  
1.US Dollar  
2.Euro  
3.Yen  
4.Exit  
Enter Choice  
**5**  
Invalid option  
1.US Dollar  
2.Euro  
3.Yen  
4.Exit  
Enter Choice  
**4**

### **Problem Requirements:**

#### Java

|  |  |  |
| --- | --- | --- |
| Keyword | Min Count | Max Count |
| HashMap | 1 | - |

HashMap - Numbers to Bin

Write a Java program to group the set of numbers into corresponding bin. The number of bins are fixed as 10,20,30,40. Input consists of set of number with in a range of 1 - 40. Classify each number and increment the corresponding bin.  
Maintain the bin's in HashMap with key as 10,20,30,40 and their corresponding integer count as value.  
  
Create a main class "Main.java"  
First input corresponds to the number of elements and followed by N number of elements  
  
Create a class **Histogram** with below specification,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Modifier** | **Attribute/Method** | **Type/Return type** | **Name** | **Description** |
| private | Attribute | HashMap<Integer,Integer> | bins | Maintains the 4 bins and their corresponding count |
| public | Method | void | addNumber(Integer) | Add the input number to corresponding bin |
| public | Method | void | displayHistogram() | Display the histogram represented by number of stars for each bin Refer the sample input/output |

Read the input numbers in the main class and call addNumber() method in the Histogram class to add the number to the corresponding bin.  
  
**Input and Output Format:**  
First input corresponds to the number of elements (N). Followed by N integers.  
Refer sample input and output for formatting specifications.  
  
**[All text in bold corresponds to input and the rest corresponds to output]**  
**Sample Input/Output :**  
**10  
1  
5  
14  
18  
21  
25  
30  
35  
19  
3**  
Histogram  
10 : \*\*\*  
20 : \*\*\*  
30 : \*\*\*  
40 : \*

**---------------------------------TYPE 4-------------------------------------------**

**Wicket details**

**[Note :  
Strictly adhere to the object oriented specifications given as a part of the problem statement.  
Use the same class names and member variable names.  
Follow the naming conventions mentioned for getters / setters.  
Create 3 separate classes in 3 separate files.]**

Create a class named **Wicket** with the following private member variables / attributes

* Long over
* Long ball
* String wicketType
* String playerName
* String bowlerName

Include appropriate**getters**, **setters** and**constructors**.

Override the t**oString()** method to display the wicket details in the format specified in the output.

Create a class named **WicketBO** and include the following methods

|  |  |  |
| --- | --- | --- |
| No | Method Name | Method Description |
| 1 | void displayAllWicketDetails(Wicket[] wicketList) | In this method, display all the wicket details. |
| 2 | void displaySpecificWicketDetails(Wicket[] wicketList, String wicketType) | In this method, display all the details of a specific wicket (wicketType). Refer sample output. |

Create another class and write a main method to test the above classes.

**Input and Output Format:**

 Refer sample input and output for formatting specifications.

Use array of objects to read wicket details and use string.Split() function to display the wicket details  
Note : The statement "**Wicket Details**" in the output is displayed in the method inside the BO class.

All text in bold corresponds to input and the rest corresponds to output.

**Sample Input and Output :**  
Enter the number of wickets  
**3**  
Enter the details of wicket 1  
**5,2,LBW,Gautam Gambir,Aswin**  
Enter the details of wicket 2  
**6,6,Bowled,Brad Hogg,Dwayne Bravo**  
Enter the details of wicket 3  
**7,3,LBW,Robin Uthappa,Ravindra Jadeja**  
Wicket Details  
Wicket 1  
Over:5  
Ball:2  
Wicket Type:LBW  
Player Name:Gautam Gambir  
Bowler Name:Aswin  
Wicket 2  
Over:6  
Ball:6  
Wicket Type:Bowled  
Player Name:Brad Hogg  
Bowler Name:Dwayne Bravo  
Wicket 3  
Over:7  
Ball:3  
Wicket Type:LBW  
Player Name:Robin Uthappa  
Bowler Name:Ravindra Jadeja  
Enter the wicket type to be searched  
**LBW**  
Wicket 1  
Over:5  
Ball:2  
Wicket Type:LBW  
Player Name:Gautam Gambir  
Bowler Name:Aswin  
Wicket 2  
Over:7  
Ball:3  
Wicket Type:LBW  
Player Name:Robin Uthappa  
Bowler Name:Ravindra Jadeja

**Player Details**

**[Note :  
Strictly adhere to the object oriented specifications given as a part of the problem statement.  
Use the same class names and member variable names.  
Follow the naming conventions mentioned for getters / setters.  
Create 3 separate classes in 3 separate files.]**

Create a class named **Player**with the following private member variables / attributes

* String name
* String country
* String skill

Include appropriate getters, setters and constructors.

Override the toString() method to display the player details in the format specified in the output.  
String.format("%-15s %-15s %-15s %-15s %-15s %s",)

Create a class named **PlayerBO** and include the following methods

|  |  |  |
| --- | --- | --- |
| No | Method Name | Method Description |
| 1 | void displayAllPlayerDetails(Player[] playerList) | In this method, display the details of all players. |
| 2 | void displaySpecificPlayerDetails(Player[] playerList, String countryName) | In this method, display all the details of players belonging to a specific country. Refer sample output. |

Create another class named **Main**and include a main method to test the above class   
  
**Input and Output Format:**

 Refer sample input and output for formatting specifications.

Use array of objects to read player details  
Note : The statement "**Player Details**" in the output is displayed in the method inside the BO class.

All text in bold corresponds to input and the rest corresponds to output.

**Sample Input and Output :**

Enter the number of players  
**3**  
Enter the player name  
**Dwayne Bravo**  
Enter the country name  
**West Indies**  
Enter the skill  
**All Rounder**  
Enter the player name  
**Chris Gayle**  
Enter the country name  
**West Indies**  
Enter the skill  
**All Rounder**  
Enter the player name  
**Virat Kohli**  
Enter the country name  
**India**  
Enter the skill  
**Batsman**  
Player Details  
Dwayne Bravo    West Indies     All Rounder      
Chris Gayle     West Indies     All Rounder      
Virat Kohli     India           Batsman          
Enter the country name for which players details to be known  
**West Indies**  
Player Details  
Dwayne Bravo    West Indies     All Rounder      
Chris Gayle     West Indies     All Rounder

**Match Details**

**[Note :  
  
  
  
Strictly adhere to the object oriented specifications given as a part of the problem statement.  
  
  
Use the same class names and member variable names  
  
  
Follow the naming conventions mentioned for getters / setters.  
  
  
Create 3 separate classes in 3 separate files.]**

Create a class named **Match** with the following private member variables / attributes

String date;

String teamOne;

String teamTwo;

String venue;

Include appropriate getters, setters and constructors.

Override the toString() method to display the match details in the format specified in the output("%-35s %-30s %-15s %s").

Create a class named Match**BO** and include the following methods

|  |  |  |
| --- | --- | --- |
| No | Method Name | Method Description |
| 1 | void displayAllMatchDetails(Match[] matchList) | In this method, display the details of all matches. |
| 2 | void displaySpecificMatchDetails(Match[] matchList, String date) | In this method, display all the details of a match held on specific date.Refer sample output. |

**Input and Output Format:**

 Refer sample input and output for formatting specifications.  
  
  
  
Note : The statement "**Match Details**" in the output is displayed in the method inside the BO class.

All text in bold corresponds to input and the rest corresponds to output.

**Sample Input and Output :**

Enter the number of matches  
**3**  
Enter match 1 details  
Enter the match date  
**21/08/2015**  
Enter the team one  
**Chennai Super Kings**  
Enter the team two  
**Mumbai Indians**  
Enter the Venue  
**Chinnaswamy stadium,Bengalore**  
Enter match 2 details  
Enter the match date  
**21/08/2015**  
Enter the team one  
**Delhi Daredevils**  
Enter the team two  
**Royal Challengers Bangalore**  
Enter the Venue  
**Feroz Shah Kotla Ground,Delhi**  
Enter match 3 details  
Enter the match date  
**22/08/2015**  
Enter the team one  
**Rajasthan Royals**  
Enter the team two  
**Kings XI Punjab**  
Enter the Venue  
**PCA Stadium,Mohali**  
Match Details  
Team 1                                              Team 2                                                 Date                     Venue  
Chennai Super Kings                 Mumbai Indians                                21/08/2015      Chinnaswamy stadium,Bengalore  
Delhi Daredevils                           Royal Challengers Bangalore    21/08/2015      Feroz Shah Kotla Ground,Delhi  
Rajasthan Royals                         Kings XI Punjab                                22/08/2015      PCA Stadium,Mohali  
Enter the date to be searhed  
**21/08/2015**  
Match Details  
Team 1                                             Team 2                                                  Date                    Venue  
Chennai Super Kings                Mumbai Indians                                21/08/2015      Chinnaswamy stadium,Bengalore  
Delhi Daredevils                          Royal Challengers Bangalore    21/08/2015      Feroz Shah Kotla Ground,Delhi

**Outcome Details**

**[Note :  
Strictly adhere to the object oriented specifications given as a part of the problem statement.  
Use the same class names and member variable names  
Follow the naming conventions mentioned for getters / setters.  
Create 3 separate classes in 3 separate files.]**

Create a class named **Outcome**with the following private member variables / attributes

* String date
* String status
* String winnerTeam
* String playerOfMatch

Include appropriate getters, setters and constructors.

Override the toString() method to display the Outcome details in the format specified in the output.

String.format("%-20s %-20s %-20s %s",)

Create a class named **OutcomeBO** and include the following methods

|  |  |  |
| --- | --- | --- |
| No | Method Name | Method Description |
| 1 | void displayAllOutcomeDetails(Outcome[] outcomeList) | In this method, display all Outcome details |
| 2 | void displaySpecificOutcomeDetails(Outcome[] outcomeList, String date) | In this method, display all the Outcome details of matches held on a particular date. Refer sample output. |

Create another class and write a main method to test the above classes.

**Input and Output Format:**

 Refer sample input and output for formatting specifications.

Use array of objects to read outcome details  
  
Note : The statement "**Outcome Details**" in the output is displayed in the method inside the BO class.

All text in bold corresponds to input and the rest corresponds to output.

**Sample Input and Output :**  
  
  
Enter the number of matches  
**3**  
Enter match 1 details  
Enter the date  
**21/08/2015**  
Enter the status  
**Won By 4 Wickets**  
Enter the winner team  
**CSK**  
Enter the player of match  
**Ashwin**  
Enter match 2 details  
Enter the date  
**28/08/2015**  
Enter the status  
**Won By 4 Runs**  
Enter the winner team  
**MI**  
Enter the player of match  
**Pollard**  
Enter match 3 details  
Enter the date  
**21/08/2015**  
Enter the status  
**Won By 57 Runs**  
Enter the winner team  
**RCB**  
Enter the player of match  
**Kohli**  
Outcome Details  
Status               Winning Team         Player Of The Match  Date  
Won By 4 Wickets     CSK                  Ashwin               21/08/2015  
Won By 4 Runs        MI                   Pollard              28/08/2015  
Won By 57 Runs       RCB                  Kohli                21/08/2015  
Enter the date to be searhed  
**21/08/2015**  
Outcome Details  
Status               Winning Team         Player Of The Match  Date  
Won By 4 Wickets     CSK                  Ashwin               21/08/2015  
Won By 57 Runs       RCB                  Kohli                21/08/2015

**Delivery Details**

**[Note :  
Strictly adhere to the object oriented specifications given as a part of the problem statement.  
Use the same class names and member variable names  
Follow the naming conventions mentioned for getters / setters.  
Create 3 separate classes in 3 separate files.]**

Create a class named **Delivery**with the following private member variables / attributes

* Long over
* Long ball
* Long runs
* String nonStriker;
* String batsman
* String bowler

Include appropriate getters, setters and constructors.

Override the toString() method to display All Delivery details in the format specified in the output.

String.format("%-20s %-20s %-20s %-20s %-20s %s",)

Create a class named **DeliveryBO** and include the following methods

|  |  |  |
| --- | --- | --- |
| No | Method Name | Method Description |
| 1 | void displayAllDeliveryDetails(Delivery[] deliveryList) | In this method, display all Delivery details |
| 2 | void displayBatsmanBowlerDetails(Delivery[] deliveryList, long ball,long over) | In this method, display batsman and bowler details who bowled and faced the ball number got from the user. Refer sample output. |
| 3 | void displayMaximumRunDetails(Delivery[] deliveryList) | In this method, display maximum runs taken and its over number and ball number details.Refer sample output. |

Create another class and write a main method to test the above classes.Use switch to drive methods.

**Input and Output Format:**

 Refer sample input and output for formatting specifications.

Use array of objects to read delivery details  
  
Note : The statement "**Delivery Details**" in the output is displayed in the method inside the BO class.

All text in bold corresponds to input and the rest corresponds to output.

**Sample Input and Output 1:**  
Enter the number of deliveries  
**4**  
Enter the over  
**1**  
Enter the ball  
**1**  
Enter the runs  
**4**  
Enter the batsman name  
**MS Dhoni**  
Enter the bowler name  
**Dwane Bravo**  
Enter the nonStriker name  
**Gayle**  
Enter the over  
**2**  
Enter the ball  
**1**  
Enter the runs  
**2**  
Enter the batsman name  
**Ashwin**  
Enter the bowler name  
**DJ Bravo**  
Enter the nonStriker name  
**Watson**  
Enter the over  
**3**  
Enter the ball  
**3**  
Enter the runs  
**6**  
Enter the batsman name  
**Murali vijay**  
Enter the bowler name  
**Dwane Bravo**  
Enter the nonStriker name  
**Gayle**  
Enter the over  
**5**  
Enter the ball  
**4**  
Enter the runs  
**4**  
Enter the batsman name  
**Kohli**  
Enter the bowler name  
**DJ Bravo**  
Enter the nonStriker name  
**Watson**  
Enter your choice  
1.View delivery details  
2.Batsman and Bowler  
3.Maximum runs  
**1**  
Delivery Details  
Over                 Ball                 Runs                 Batsman              Bowler                        NonStriker  
1                         1                   4                       MS Dhoni              Dwane Bravo          Gayle  
2                       1                   2                       Ashwin                   DJ Bravo                   Watson  
3                         3                  6                       Murali vijay           Dwane Bravo          Gayle  
5                          4                  4                       Kohli                         DJ Bravo                   Watson  
  
  
**Sample Input and Output 2:**  
  
Enter the number of deliveries  
**4**  
Enter the over  
**1**  
Enter the ball  
**1**  
Enter the runs  
**4**  
Enter the batsman name  
**MS Dhoni**  
Enter the bowler name  
**Dwane Bravo**  
Enter the nonStriker name  
**Gayle**  
Enter the over  
**2**  
Enter the ball  
**1**  
Enter the runs  
**2**  
Enter the batsman name  
**Ashwin**  
Enter the bowler name  
**DJ Bravo**  
Enter the nonStriker name  
**Watson**  
Enter the over  
**3**  
Enter the ball  
**3**  
Enter the runs  
**6**  
Enter the batsman name  
**Murali vijay**  
Enter the bowler name  
**Dwane Bravo**  
Enter the nonStriker name  
**Gayle**  
Enter the over  
**5**  
Enter the ball  
**4**  
Enter the runs  
**4**  
Enter the batsman name  
**Kohli**  
Enter the bowler name  
**DJ Bravo**  
Enter the nonStriker name  
**Watson**  
Enter your choice  
1.View delivery details  
2.Batsman and Bowler  
3.Maximum runs  
**2**  
Enter the over for which batsman and bowler to be known  
**4**  
Enter the ball for which batsman and bowler to be known  
**4**  
Batsman : Kohli  
Bowler :DJ Bravo  
  
**Sample Input and Output 3:**  
Enter the number of deliveries  
**4**  
Enter the over  
**1**  
Enter the ball  
**1**  
Enter the runs  
**4**  
Enter the batsman name  
**MS Dhoni**  
Enter the bowler name  
**Dwane Bravo**  
Enter the nonStriker name  
**Gayle**  
Enter the over  
**2**  
Enter the ball  
**1**  
Enter the runs  
**2**  
Enter the batsman name  
**Ashwin**  
Enter the bowler name  
**DJ Bravo**  
Enter the nonStriker name  
**Watson**  
Enter the over  
**3**  
Enter the ball  
**3**  
Enter the runs  
**6**  
Enter the batsman name  
**Murali vijay**  
Enter the bowler name  
**Dwane Bravo**  
Enter the nonStriker name  
**Gayle**  
Enter the over  
**5**  
Enter the ball  
**4**  
Enter the runs  
**4**  
Enter the batsman name  
**Kohli**  
Enter the bowler name  
**DJ Bravo**  
Enter the nonStriker name  
**Watson**  
Enter your choice  
1.View delivery details  
2.Batsman and Bowler  
3.Maximum runs  
**3**  
Maximum Runs : 6  
Over : 3  
Ball : 3

-**---------------------------------TYPE 5--------------------**

Problem Description

**Trainee Class**

**[Note :**

**Strictly adhere to the object oriented specifications given as a part of the problem statement.**

**Download and use the TEMPLATE CODE provided here.**

**Use the same class names and member variable names.**

**Create 4 separate classes in 4 separate files.]**

Create a class named **Trainee** with the following private member variables

|  |  |
| --- | --- |
| Data Type | Variable Name |
| int | id |
| String | name |
| String | batchName |
| int | age |

Include appropriate getters and setters.

Include a 4 argument constructor. The order of parameters passed to the constructor is id, name, batchName, age.

Override the toString (Java) method to display the trainee details in the given format.

id name batchname age

[ All details are separated by a single space. Don't include a new line in the string returned from this method. Refer sample output]

Create a class named **IdRangeException** with one member variable msg of type String.

Include a default constructor and 1-argument constructor.

Override the toString method to display the exception details in this format:

IdRangeException: msg

This exception is thrown when the trainee id is not a 5-digit number.

Create a class named **AgeRangeException** with one member variable msg of type String.

Include a default constructor and 1-argument constructor.

Override the toString method to display the exception details in this format:

AgeRangeException: msg

This exception is thrown when the age of the trainee is less than 20 or greater than 25.

When the user doesn't enter an integer as input to the trainee id / age, a Number Format Exception is thrown.

Whenever an exception occurs, the user will be asked to reenter the details till he enters the correct details.

Refer sample output for formatting details.

Create a class called Main to test the Trainee class and the various exceptions.

**Input and Output Format :**

Refer sample input and output for formatting specifications.

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

**SAMPLE INPUT AND OUTPUT 1**  
Enter the id of the trainee  
**1**  
IdRangeException: Id not within range  
Please enter the id again  
**100000**  
IdRangeException: Id not within range  
Please enter the id again  
**10000**  
Enter the name of the trainee  
**Ananthi**  
Enter the batch name of the trainee  
**PUN21DN003**  
Enter the age of the trainee  
**21**  
The trainee details are  
10000 Ananthi PUN21DN003 21  
  
**SAMPLE INPUT AND OUTPUT 2**  
Enter the id of the trainee  
**A1**  
java.lang.NumberFormatException: For input string: "A1"  
Please enter the id again  
**1**  
IdRangeException: Id not within range  
Please enter the id again  
**10000**  
Enter the name of the trainee  
**Priya**  
Enter the batch name of the trainee  
**CBE11AJ001**  
Enter the age of the trainee  
**23**  
The trainee details are  
10000 Priya CBE11AJ001 23  
  
**SAMPLE INPUT AND OUTPUT 3**  
Enter the id of the trainee  
**56578**  
Enter the name of the trainee  
**Haritha**  
Enter the batch name of the trainee  
**BGL13DW009**  
Enter the age of the trainee  
**19**  
AgeRangeException: Age not within range  
Please enter the age again  
**27**  
AgeRangeException: Age not within range  
Please enter the age again  
**22**  
The trainee details are  
56578 Haritha BGL13DW009 22  
  
**SAMPLE INPUT AND OUTPUT 4**  
Enter the id of the trainee  
**89334**  
Enter the name of the trainee  
**Pragha**  
Enter the batch name of the trainee  
**KOL11DN025**  
Enter the age of the trainee  
**ww**  
java.lang.NumberFormatException: For input string: "ww"  
Please enter the age again  
**22**  
The trainee details are  
89334 Pragha KOL11DN025 22

Problem Description

**Batch Class**

**[Note :**

**Strictly adhere to the object oriented specifications given as a part of the problem statement.**

**Download and use the TEMPLATE CODE provided here.**

**Use the same class names and member variable names.**

**Create 4 separate classes in 4 separate files.]**

Create a class named Batch with the following private member variables

|  |  |
| --- | --- |
| Data Type | Variable Name |
| String | name |
| String | trainerName |
| int | batchSize |
| int | averageMarks |

Include appropriate getters and setters.

Include a 4 argument constructor. The order of parameters passed to the constructor is name, trainer name, batch size and average marks.

Override the toString (Java) / ToString (C#) method to display the batch details in the given format.

name trainername batchsize averagemarks

[ All details are separated by a single space. Don't include a new line in the string returned from this method. Refer sample output]

Create a class named **BatchSizeException** with one member variable msg of type String.

Include a default constructor and 1-argument constructor.

Override the toString method to display the exception details in this format:

BatchSizeException: msg

This exception is thrown when the batch size is less than 10 or greater than 30.

Create a class named **AverageMarksException** with one member variable msg of type String.

Include a default constructor and 1-argument constructor.

Override the toString method to display the exception details in this format:

AverageMarksException: msg

This exception is thrown when the average marks is less than 0 or greater than 100.

When the user doesn't enter an integer as input to the batch size or average marks, a Number Format Exception is thrown.

Whenever an exception occurs, the user will be asked to reenter the details till he enters the correct details.

Refer sample output for formatting details.

Create a class called Main to test the Batch class and the various exceptions.

**Input and Output Format :**

Refer sample input and output for formatting specifications.

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

**SAMPLE INPUT AND OUTPUT 1**  
Enter the name of the batch  
**PUN15DN019**  
Enter the trainer  name of the batch  
**Ananthi**  
Enter the size of the batch  
**5**  
BatchSizeException: Batch size not within range  
Please enter the batch size again  
**35**  
BatchSizeException: Batch size not within range  
Please enter the batch size again  
**30**  
Enter the average marks of the batch  
**25**  
The batch details are  
PUN15DN019 Ananthi 30 25  
  
**SAMPLE INPUT AND OUTPUT 2**  
Enter the name of the batch  
**PUN16DN021**  
Enter the trainer  name of the batch  
**Haritha**  
Enter the size of the batch  
**q**  
java.lang.NumberFormatException: For input string: "q"  
Please enter the batch size again  
**9**  
BatchSizeException: Batch size not within range  
Please enter the batch size again  
**15**  
Enter the average marks of the batch  
**120**  
AverageMarkException: Average marks not within range  
Please enter the average marks again  
**50**  
The batch details are  
PUN16DN021 Haritha 15 50  
  
**SAMPLE INPUT AND OUTPUT 3**  
Enter the name of the batch  
**CBE10AJ015**  
Enter the trainer  name of the batch  
**Priya**  
Enter the size of the batch  
**q**  
java.lang.NumberFormatException: For input string: "q"  
Please enter the batch size again  
**12**  
Enter the average marks of the batch  
**60**  
The batch details are  
CBE10AJ015 Priya 12 60  
  
**SAMPLE INPUT AND OUTPUT 4**  
Enter the name of the batch  
**BGL15DW017**  
Enter the trainer  name of the batch  
**Madhu**  
Enter the size of the batch  
**21**  
Enter the average marks of the batch  
**-1**  
AverageMarkException: Average marks not within range  
Please enter the average marks again  
**120**  
AverageMarkException: Average marks not within range  
Please enter the average marks again  
**100**  
The batch details are  
BGL15DW017 Madhu 21 100

Problem Description

**Trainer Class**

**[Note :**

**Strictly adhere to the object oriented specifications given as a part of the problem statement.**

**Download and use the TEMPLATE CODE provided here.**

**Use the same class names and member variable names.**

**Create 4 separate classes in 4 separate files.]**

Create a class named **Trainer** with the following private member variables

|  |  |
| --- | --- |
| Data Type | Variable Name |
| int | id |
| String | name |
| String | stream |
| int | noOfBatches |

Include appropriate getters and setters.

Include a 4 argument constructor. The order of parameters passed to the constructor is id, name, stream, number of batches.

Override the toString (Java) / ToString (C#) method to display the trainer details in the given format.

id name stream noofbatches

[ All details are separated by a single space. Don't include a new line in the string returned from this method. Refer sample output]

Create a class named **IdRangeException** with one member variable msg of type String.

Include a default constructor and 1-argument constructor.

Override the toString method to display the exception details in this format:

IdRangeException: msg

This exception is thrown when the trainee id is not a 5-digit number.

Create a class named **InvalidStreamException** with one member variable msg of type String.

Include a default constructor and 1-argument constructor.

Override the toString method to display the exception details in this format:

InvalidStreamException: msg

This exception is thrown when the stream is not one of the following --- Java, CSharp, Mainframes

When the user doesn't enter an integer as input to the trainee id , a Number Format Exception is thrown.

Whenever an exception occurs, the user will be asked to reenter the details till he enters the correct details.

Refer sample output for formatting details.

Create a class called Main to test the Trainer class and the various exceptions.

**Input and Output Format :**

Refer sample input and output for formatting specifications.

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

**SAMPLE INPUT AND OUTPUT 1**  
Enter the id of the trainer  
**1**     
IdRangeException: Id not within range  
Please enter the id again  
**99991**  
Enter the name of the trainer  
**Ananthi**  
Enter the stream of the trainer  
**Csharp**  
InvalidStreamException: An invalid stream  
Please enter the stream again  
**CSharp**  
Enter the number of batches allotted  
**23**  
The trainer details are  
99991    Ananthi              CSharp               23  
  
  
**SAMPLE INPUT AND OUTPUT 2**  
Enter the id of the trainer  
**99999999**  
IdRangeException: Id not within range  
Please enter the id again  
**192892**  
IdRangeException: Id not within range  
Please enter the id again  
**45456**  
Enter the name of the trainer  
**Haritha**  
Enter the stream of the trainer  
**Java**  
Enter the number of batches allotted  
**25**  
The trainer details are  
45456    Haritha              Java                 25  
  
**SAMPLE INPUT AND OUTPUT 3**  
Enter the id of the trainer  
**q**  
java.lang.NumberFormatException: For input string: "q"  
Please enter the id again  
**46787**  
Enter the name of the trainer  
**Madhu**  
Enter the stream of the trainer  
**Mainframes**  
Enter the number of batches allotted  
**38**  
The trainer details are  
46787    Madhu                Mainframes           38  
  
  
**SAMPLE INPUT AND OUTPUT 4**  
Enter the id of the trainer  
**78787**  
Enter the name of the trainer  
**Pragha**  
Enter the stream of the trainer  
**Sql**        
InvalidStreamException: An invalid stream  
Please enter the stream again  
**CSharp**  
Enter the number of batches allotted  
**26**  
The trainer details are  
78787    Pragha               CSharp               26

**BatchOwner Class**

**[Note :**

**Strictly adhere to the object oriented specifications given as a part of the problem statement.**

**Download and use the TEMPLATE CODE provided here.**

**Use the same class names and member variable names.**

**Create 4 separate classes in 4 separate files.]**

Create a class named BatchOwner with the following private member variables

|  |  |
| --- | --- |
| Data Type | Variable Name |
| int | id |
| String | name |
| String | location |
| int | noOfBatches |

Include appropriate getters and setters.

Include a 4 argument constructor. The order of parameters passed to the constructor is id, name, location, number of batches.

Override the toString (Java) / ToString (C#) method to display the batch owner details in the given format.

id name location noofbatches

[ All details are separated by a single space. Don't include a new line in the string returned from this method. Refer sample output]

Create a class named **IdRangeException** with one member variable msg of type String.

Include a default constructor and 1-argument constructor.

Override the toString method to display the exception details in this format:

IdRangeException: msg

This exception is thrown when the trainee id is not a 5-digit number.

Create a class named **AllottedBatchesException** with one member variable msg of type String.

Include a default constructor and 1-argument constructor.

Override the toString method to display the exception details in this format:

AllottedBatchesException: msg

This exception is thrown when the the number of batches allotted is less than 5 or greater than 10.

When the user doesn't enter an integer as input to the id / number of batches allotted, a Number Format Exception is thrown.

Whenever an exception occurs, the user will be asked to reenter the details till he enters the correct details.

Refer sample output for formatting details.

Create a class called Main to test the BatchOwner class and the various exceptions.

**Input and Output Format :**

Refer sample input and output for formatting specifications.

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

**Sample input and output 1**  
Enter the id of the batch owner  
**1**  
IdRangeException: Id not within range  
Please enter the id again  
**1000200**  
IdRangeException: Id not within range  
Please enter the id again  
**98919**  
Enter the name of the batch owner  
**Ananthi**  
Enter the location of the batch owner  
**Pune**  
Enter the number of batches allotted  
**8**  
The batch owner details are  
98919 Ananthi Pune 8  
  
**sample input and output 2**  
Enter the id of the batch owner  
**65656**  
Enter the name of the batch owner  
**Haritha**  
Enter the location of the batch owner  
**Chennai**  
Enter the number of batches allotted  
**14**  
AllottedBatchesException: Number of batches allotted not within range  
Please enter the number of batches allotted again  
**6**  
The batch owner details are  
65656 Haritha Chennai 6  
  
  
**sample input and output 3**  
Enter the id of the batch owner  
**1**  
IdRangeException: Id not within range  
Please enter the id again  
**a**  
java.lang.NumberFormatException: For input string: "a"  
Please enter the id again  
**13134**  
Enter the name of the batch owner  
**Pragha**  
Enter the location of the batch owner  
**Coimbatore**  
Enter the number of batches allotted  
**w**  
java.lang.NumberFormatException: For input string: "w"  
Please enter the number of batches allotted again  
**4**  
AllottedBatchesException: Number of batches allotted not within range  
Please enter the number of batches allotted again  
**7**  
The batch owner details are  
13134 Pragha Coimbatore 7  
  
**sample input and output 4**  
Enter the id of the batch owner  
**ad111**  
java.lang.NumberFormatException: For input string: "ad111"  
Please enter the id again  
**45678**  
Enter the name of the batch owner  
**Madhu**  
Enter the location of the batch owner  
**Kolkata**  
Enter the number of batches allotted  
**5**  
The batch owner details are  
45678 Madhu Kolkata 5

**Course Class**

**[Note :**

**Strictly adhere to the object oriented specifications given as a part of the problem statement.**

**Download and use the TEMPLATE CODE provided here.**

**Use the same class names and member variable names.**

**Create 4 separate classes in 4 separate files.]**

Create a class named Course with the following private member variables

|  |  |
| --- | --- |
| Data Type | Variable Name |
| int | id |
| String | name |
| String | type |
| int | duration |

Include appropriate getters and setters.

Include a 4 argument constructor. The order of parameters passed to the constructor is id, name, type, duration.

Override the toString (Java) / ToString (C#) method to display the course details in the given format.

id name type duration

[ All details are separated by a single space. Don't include a new line in the string returned from this method. Refer sample output]

Create a class named **IdRangeException** with one member variable msg of type String.

Include a default constructor and 1-argument constructor.

Override the toString method to display the exception details in this format:

IdRangeException: msg

This exception is thrown when the trainee id is not a 5-digit number.

Create a class named **InvalidTypeException** with one member variable msg of type String.

Include a default constructor and 1-argument constructor.

Override the toString method to display the exception details in this format:

InvalidTypeException: msg

This exception is thrown when the type is not one of the following --- HandsOn, Theory, SelfLearning

When the user doesn't enter an integer as input to the trainee id , a Number Format Exception is thrown.

Whenever an exception occurs, the user will be asked to reenter the details till he enters the correct details.

Refer sample output for formatting details .

Create a class called Main to test the Course class and the various exceptions.

**Input and Output Format :**

Refer sample input and output for formatting specifications.

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

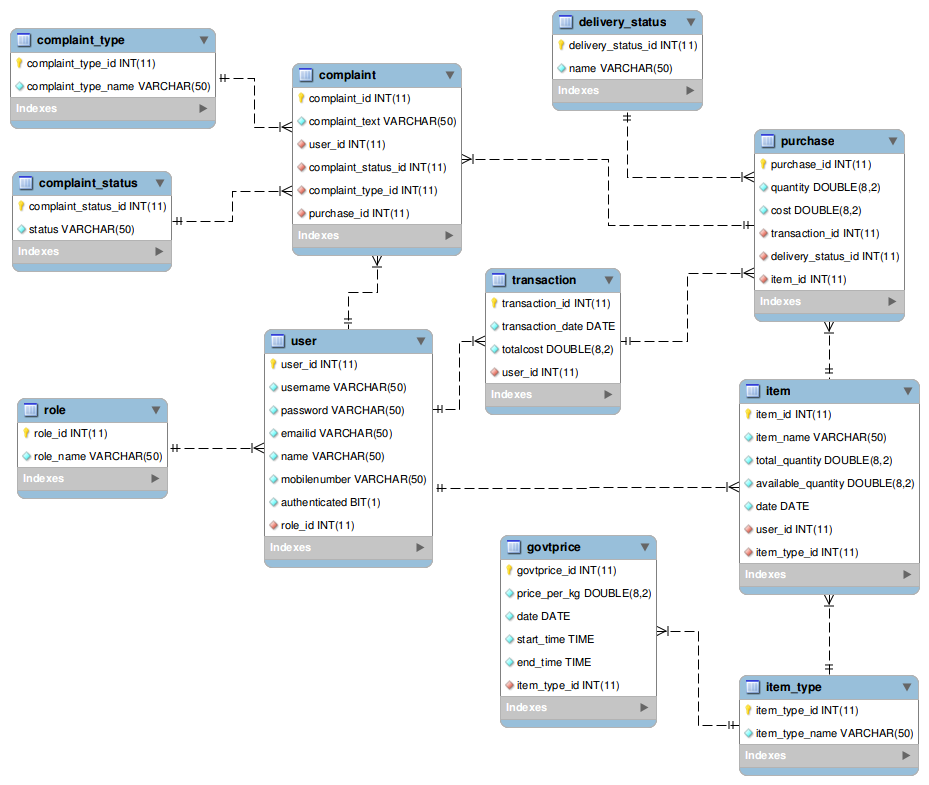
**SAMPLE INPUT AND OUTPUT 1**  
Enter the id of the course  
**1**  
IdRangeException: Id not within range  
Please enter the id again  
**45345**    
Enter the name of the course  
**Java**  
Enter the type of the course  
**Handson**  
InvalidTypeException: Invalid Type  
Please enter the type again  
**HandsOn**  
Enter the duration of the course  
**30**  
The course details are  
45345 Java HandsOn 30  
  
**SAMPLE INPUT AND OUTPUT 2**  
Enter the id of the course  
**A**  
java.lang.NumberFormatException: For input string: "A"  
Please enter the id again  
**12123**  
Enter the name of the course  
**Dotnet**  
Enter the type of the course  
**Assessment**  
InvalidTypeException: Invalid Type  
Please enter the type again  
**Theory**  
Enter the duration of the course  
**120**  
The course details are  
12123 Dotnet Theory 120  
  
**SAMPLE INPUT AND OUTPUT 3**  
Enter the id of the course  
**45456**  
Enter the name of the course  
**Mainframe**  
Enter the type of the course  
**Quiz**  
InvalidTypeException: Invalid Type  
Please enter the type again  
**SelfLearning**  
Enter the duration of the course  
**45**  
The course details are  
45456 Mainframe SelfLearning 45  
  
  
**SAMPLE INPUT AND OUTPUT 4**  
Enter the id of the course  
**233455**  
IdRangeException: Id not within range  
Please enter the id again  
**23456**  
Enter the name of the course  
**Java**  
Enter the type of the course  
**SelfLearning**  
Enter the duration of the course  
**60**  
The course details are  
23456 Java SelfLearning 60

**---------------------------------TYPE 6-------------------------------------------**

JDBC - Update User Contact

Write a java program to list all the available users in the table ordered based on userId. Get the username input from the user which needs to be updated and get the new contact information (mobile number and email) from the user and update the details into the database.  
  
**Specification:**  
Create **Main.java** with main method  
  
Create **User.java** domain class with below attributes,

* userId - Long
* username - String
* password - String
* emailId - String
* name- String
* mobileNumber - String
* authenticated - Boolean

Include getter and setter method for all the attributes  
Include a constructor with below arguments,  
**public User(userId,username,password,emailId,name,mobileNumber,authenticated)**  
  
Create **UserDAO** with below methods,  
**public List<User> getAllUsers()** - Method used to get all the users from the database  
**public void updateContactDetails(String username,String mobilenumber,String email)** - Method used to update the user's contact details (Mobile number and Email)  
  
**DBUtil** class is used to create and return new database connection. This utility class is used by DAO classes to get the new connection.  
  
**Schema Diagram:**  


**Note:**  
Use the below code to retrieve the connection details from mysql properties to establish connection

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

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

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

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

**mysql.properties :**

Use the following code to establish connection.

db.url=jdbc:mysql://localhost:3306/jdbc

db.username=root

db.password=test

**Sample Input and Output:**  
User List  
User ID         Username        Email ID                       Name            Mobilenumber    Authenticated    
1               madhu           madhu@gmail.com                madhu           8012479852      false            
2               sri shobi       shobi@gmail.com                sri             7981173215      false            
Enter the username you want update  
**madhu**  
Enter the new mobilenumber  
**987456210**  
Enter the new email ID  
**madhul@gmail.com**  
Updated user list  
User ID         Username        Email ID                                    Name                    Mobilenumber    Authenticated    
1                          madhu           madhul@gmail.com               madhu                    987456210       false            
2                           sri shobi       shobi@gmail.com                     sri                            7981173215      false

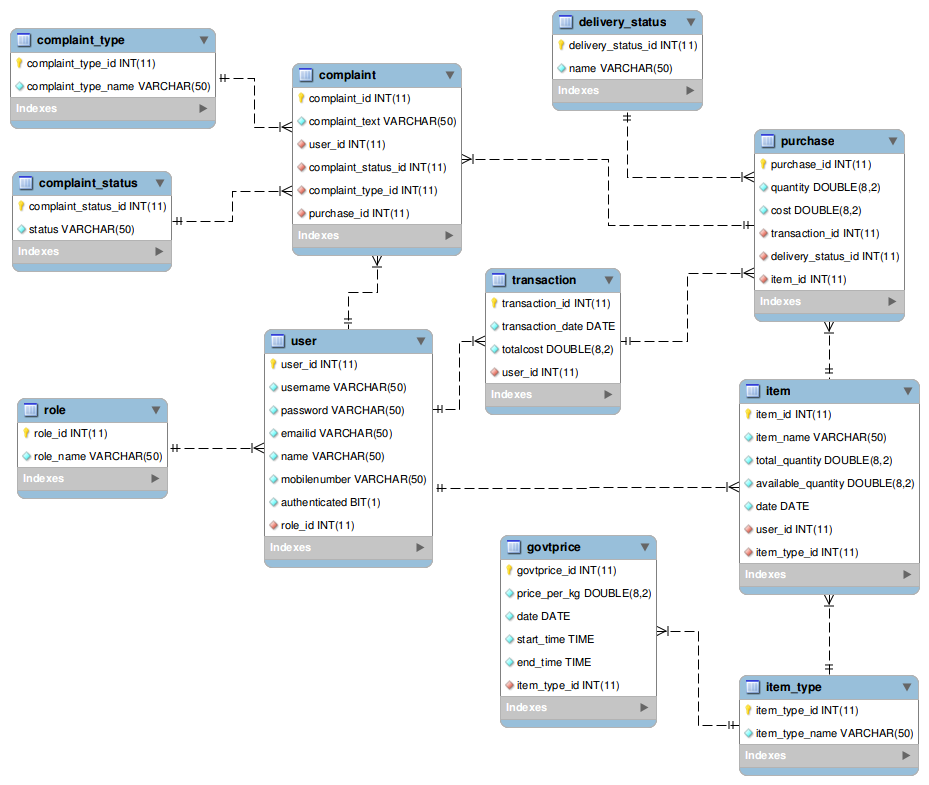
JDBC - User Approve/Deny

Write a java program to list all the available users in the table ordered based on userId. Get the username input from the user which needs to be approved/denied and update the corresponding status in authenticated flag into the database.  
  
**Specification:**  
Create **Main.java** with main method  
  
Create **User.java** domain class with below attributes,

* userId - Long
* username - String
* password - String
* emailId - String
* name- String
* mobileNumber - String
* authenticated - Boolean

Include getter and setter method for all the attributes  
Include a constructor with below arguments,  
**public User(userId,username,password,emailId,name,mobileNumber,authenticated)**  
  
Create **UserDAO** with below methods,  
**public List<User> getAllUsers()** - Method used to get all the users from the database  
**public void updateUserDetails(String username,Boolean authenticated)** - Method used to updated the authenticated flag. True value indicates an approved user.  
  
**DBUtil** class is used to create and return new database connection. This utility class is used by DAO classes to get the new connection.  
  
**Output display format:**  
For display users details

System.out.format("%-15s %-15s %-30s %-15s %-15s\n","User ID","Username","Email ID","Mobilenumber","Authenticated");

**Schema Diagram :**  
  
  
  
  
**mysql.properties :**  
db.url=jdbc:mysql://localhost:3306/jdbc  
db.username=root  
db.password=test  
  
**Note:**  
Use the below code to retrieve the connection details from mysql.properties to establish connection  
         ResourceBundle rb= ResourceBundle.getBundle("mysql");  
        String url=rb.getString("db.url");  
        String user=rb.getString("db.username");  
        String pass=rb.getString("db.password");  
  
**Sample Input and Output:**

User List

User ID         Username        Email ID                            Mobilenumber                Authenticated

1                   madhu              madhul@gmail.com          987456210                     Denied

2                   sri shobi           shobi@gmail.com              7981173215                   Denied

Enter the username you want to (Approve/Deny)

**madhu**

Enter Approve/Deny

**Approve**

Updated user list

User ID         Username        Email ID                                    Mobilenumber             Authenticated

1                   madhu                 madhul@gmail.com              987456210                   Approved

2                   sri shobi              shobi@gmail.com                   7981173215                Denied

JDBC - Change Order Status

Write a java program to list all available orders from the table ordered based on purchaseId. Get the purchase id input from the user and update the status into the database.  
  
**Specification:**  
Create **Main.java** with main method  
Create **Purchase.java** domain class with below attributes,

* purchaseId - Long
* item - Item
* quantity - Double
* cost - Double
* transaction - Transaction
* deliveryStatus - DeliveryStatus

Include getter and setter method for all the attributes  
Include a constructor with below arguments,  
**public Purchase(item, quantity, cost, transaction, deliveryStatus)  
public Purchase(purchaseId, item, quantity, cost, transaction, deliveryStatus)**  
  
Create **PurchaseDAO** with below methods,  
**public List<Purchase> getAllPurchase()** - Method used to get all the purcahses from the database, All associated objects like transaction and deliveryStatus should be populated.  
**public List<DeliveryStatus> getAllStatus()** - Get all delivery status from the database order by status id  
**public void updateDeliveryStatus(Long purchase\_id,int delivery\_status\_id)** - Method used to update the delivery status of a purchase.  
  
Create **DeliveryStatus** domain class with below attributes,

* deliveryStatusId - Long
* name - String

Include getter and setter method for all the attributes  
Include a constructor with below arguments,  
public DeliveryStatus(Long deliveryStatusId, String name)  
  
Create **Transaction** domain class with below attributes,

* transactionId - Long
* transactionDate - Date
* user - User
* totalcost - Double
* Include getter and setter method for all the attributes

Include a constructor with below arguments,  
**public Transaction(Date transactionDate, User user)**  
  
Create **User** domain class with below attributes,

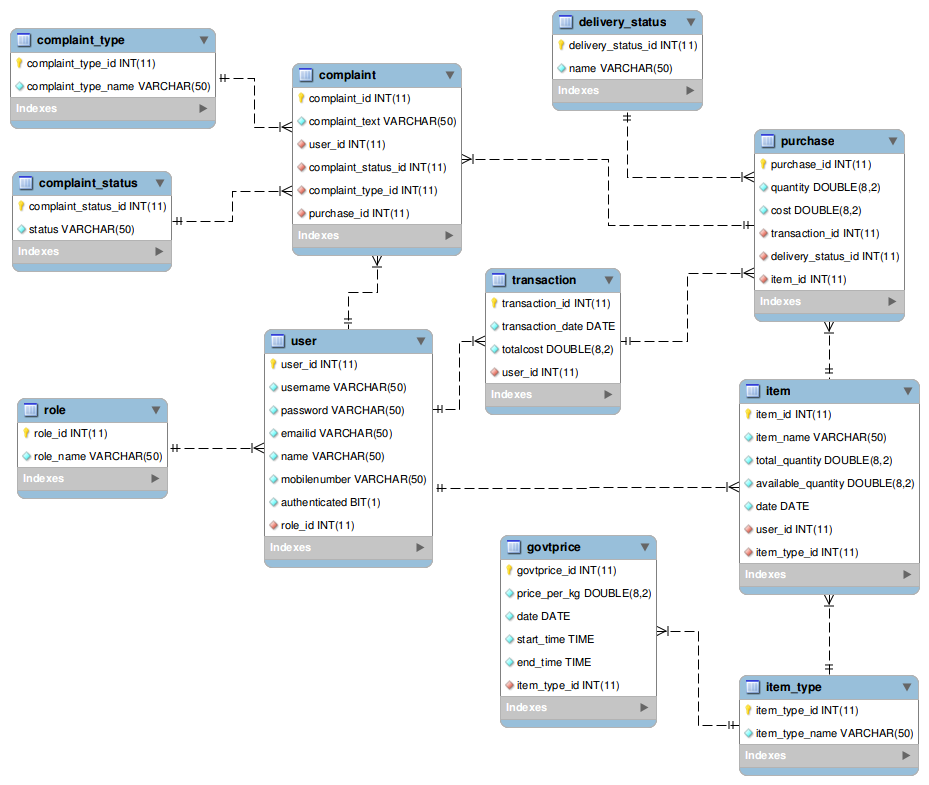
* userId - Long
* username - String
* password - String
* emailId - String
* name - String
* mobileNumber - String
* authenticated - Boolean

Include getter and setter method for all the attributes  
Include constructors with below arguments,  
**public User(int userId, String username, String emailid,String mobilenumber, Boolean authenticated)  
public User(String username,String mobileNumber)**  
  
Create **Item** domain class with below attributes,

* itemId - Long
* user - User
* itemType - ItemType
* itemName - String
* totalQuantity - Double
* availableQuantity - Double
* date - Date

Include getter and setter method for all the attributes  
Include a constructor with below arguments,  
**public Item(User user, String itemName)**  
  
Create **ItemType** domain class with below attributes,

* itemType - Long
* itemTypeName - String

Include getter and setter method for all the attributes  
Include a constructor with below arguments,  
**public ItemType(int itemType, String itemTypeName)**  
  
DBUtil class is used to create and return new database connection. This utility class is used by DAO classes to get the new connection.  
  
  
**Schema Diagram :**  
  
  
**mysql.properties :**  
db.url=jdbc:mysql://localhost:3306/jdbc  
db.username=root  
db.password=test  
  
**Note:**  
Use the below code to retrieve the connection details from mysql.properties to establish connection  
         ResourceBundle rb= ResourceBundle.getBundle("mysql");  
        String url=rb.getString("db.url");  
        String user=rb.getString("db.username");  
        String pass=rb.getString("db.password");  
  
  
**Sample Input and Output:**

Order list

Purchase Id     Date                         Item Name          Quantity          Cost               Username         Mobile Number                Delivery Status

1                        12-07-2015             Beetroot               50.0                   250.0             madhu              7865435643                       Awaiting for Approval

2                         22-07-2015           Cabbage             15.0                   300.0              maddy              7004332661                       Delivered

3                         21-03-2016           Barley                   100.0                 2000.0           vysnavi              8967543234                       Processing

Enter the purchase id to change the delivery status

**1**

Status list

Status Id       Name

1                      Approval

2                     Processing

3                     Delivered

4                     Out for Delivery

5                     Returned

6                     Awaiting for Approval

7                     Cancelled

Enter the status id

**2**

Updated order list

Purchase Id     Date                      Item Name          Quantity        Cost               Username        Mobile Number            Delivery Status

1                         12-07-2015          Beetroot              50.0                250.0              madhu               7865435643                  Processing

2                          22-07-2015        Cabbage            15.0                300.0              maddy               7004332661                   Delivered

3                          21-03-2016         Barley                 100.0             2000.0             vysnavi               8967543234                  Processing

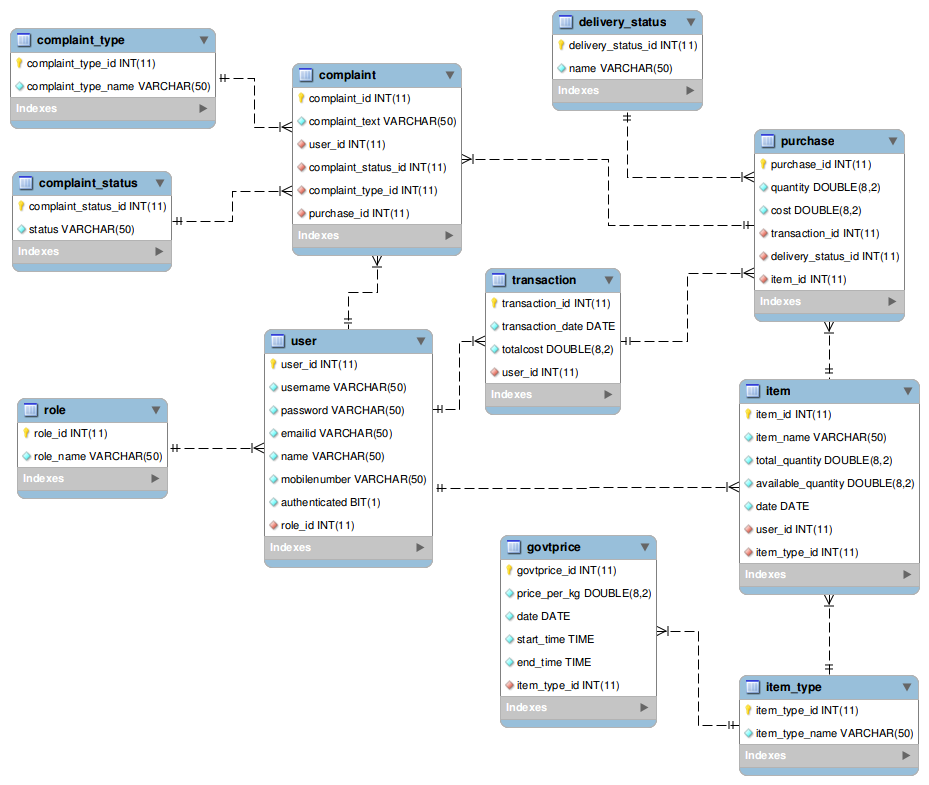
JDBC - Order Search

Write a java program to search and display all orders within the date range provided by the user.  
  
**Specification:**  
Create **Main.java** with main method  
Create **Transaction** domain class with below attributes,

* id - Long
* date - Date
* totalCost - Double
* userInstance - User

Include getter and setter method for all the attributes  
Include a constructor with below arguments,  
**public Transaction(Date date,Double totalCost,User userInstance)**  
  
Create **User** domain class with below attributes,

* id - Long
* name - String
* email - String
* username - String
* password - String
* mobileNumber - String

Include getter and setter method for all the attributes  
Include a constructor with below arguments,  
**public User(String name,String email,String userName,String password,String mobileNumber)**  
  
Create **TransactionDAO** with the below methods,  
**List<Transaction> listOrders(Date startDate,Date endDate)** - Method used to fetch the transaction from the database based on the start and end date passed in the parameter.  
  
**Schema Diagram :**  
  
  
**mysql.properties :**  
db.url=jdbc:mysql://localhost:3306/jdbc  
db.username=root  
db.password=test  
  
**Note:**  
Use the below code to retrieve the connection details from mysql.properties to establish connection  
         ResourceBundle rb= ResourceBundle.getBundle("mysql");  
        String url=rb.getString("db.url");  
        String user=rb.getString("db.username");  
        String pass=rb.getString("db.password");  
  
  
**Sample Input and Output:**

Enter the StartDate:

**2015-11-01**

Enter the EndDate:

**2015-12-31**

Transaction Date  TotalCost            User                     Email        UserName    MobileNumber

2015-11-12          720.00           madhu           madhu@gmail.com           madhu      7865435643

2015-12-12          500.00            arun            arun@gmail.com            arun      9078563454

JDBC - Raise Complaint

Write a java program to raise complaint for a purchase. Get the transaction id, purchase id and user id as input from the user. List all the transaction available in the database and then prompt the user for transaction id. Display all the purchases for the transaction. Based on the displayed list user provides the purchase id and complaint infomation as shown in the sample input and output. Use PreparedStatement for all the queries.  
  
**Specification:**  
Create **Main.java** with main method  
Create **Transaction** domain class with below attributes,

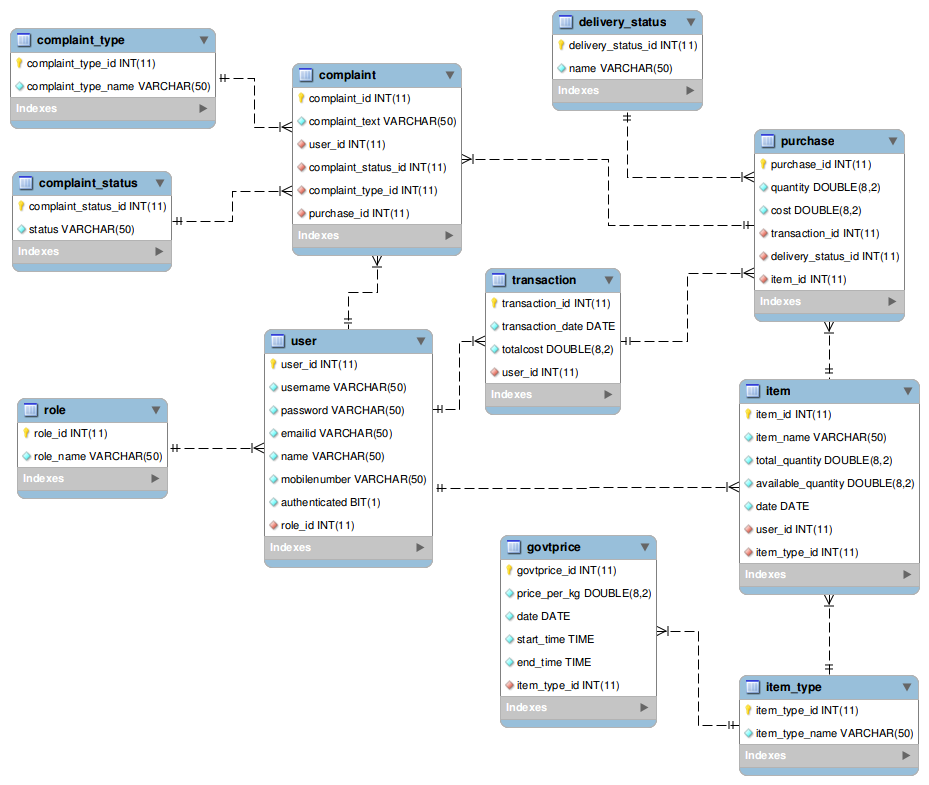
* transactionId - Long
* date - Date
* totalCost - Double
* userInstance - User

Include getter and setter method for all the attributes  
Include a constructor with below arguments,  
**public Transaction(int transactionId,Date date,Double totalCost,User userInstance)**  
  
Create **Purchase** domain class with below attributes,

* purchaseId - Long
* quantity - Double
* cost - Double
* transactionInstance - Transaction

Include getter and setter method for all the attributes  
Include a constructor with below arguments,  
**public Purchase(int purchaseId, Double quantity, Double cost, Transaction transactionInstance)**  
  
Create **Complaint** domain class with below attributes,

* complaintId - Long
* complaintTypeId - Integer
* complaintText - String
* complaintTypeName - String

Create **TransactionDAO** class with below methods,  
**public List<Transaction> listAllOrders()** - Method used to get all the transactions from the database and returns the list  
**List<Purchase> listPurchases(int transactionId)** - Method used to get all the purchase for a transaction passed in the parameter  
**public void raiseComplaint(int userId,int complaintTypeId,String complaintText,int purchaseId)** - Method used to store the compliant details for the purchase into the database  
**List<Complaint> listComplaints(int purchaseId)**- Method used to list all the complaints for the purchase passed in the parameter  
  
**Schema Diagram :**  
  
  
**mysql.properties :**  
db.url=jdbc:mysql://localhost:3306/jdbc  
db.username=root  
db.password=test  
  
**Note:**  
Use the below code to retrieve the connection details from mysql.properties to establish connection  
         ResourceBundle rb= ResourceBundle.getBundle("mysql");  
        String url=rb.getString("db.url");  
        String user=rb.getString("db.username");  
        String pass=rb.getString("db.password");  
  
  
  
**Sample input and output:**

List of Orders made:

Transaction Id    Transaction Date      TotalCost         User Id            User             Email                               UserName               Password          MobileNumber

1                          2015-07-12              600.00               3                   vysu          vysnavi@gmail.com            vysnavi                    rgvsynavi           8967543234

2                          2015-07-22              800.00               7                  madhan      mk@gmail.com                   maddy12                maddy21           7004332661

3                          2015-07-12             2000.00               2                  anandh       anandhi@gmail.com          anandhapriya         anandhi             7865435643

4                         2015-07-12              270.00                6                  madhu         madhu@gmail.com            madhu                    madhu                7865435643

5                         2015-06-22             6000.00               4                  kaushi          kaushi@gmail.com             kaushika               kaushika              8645674323

6                        2016-03-21             2000.00                3                  vysu             vysnavi@gmail.com           vysnavi                 rgvsynavi            8967543234

7                        2015-07-12             200.00                 8                   jimmy               jimesh@gmail.com        jimesh                   jie2sh                   9865425349

8                        2016-03-22             2000.00               6                   madhu         madhu@gmail.com             madhu                 madhu                  7865435643

9                        2016-08-21            7000.00                3                   vysu            vysnavi@gmail.com             vysnavi              rgvsynavi               8967543234

10                      2016-01-11            3500.00               10                  priya           priya@gmail.com                  priya                   priya                       9876545678

11                      2015-07-12            500.00                11                  arun            arun@gmail.com                 arun                    aaa                        9078563454

12                     2015-08-12            120.00                 12                  raja                 raj@gmail.com                               Raja                         ra12a                      9078664323

13                    2015-09-21              800.00                10                  priya               priya@gmail.com                           priya                          priya                      9876545678

14                    2015-06-26            1200.00                 7                  madhan            mk@gmail.com                            maddy12                    maddy21                  7004332661

15                    2015-01-12             2300.00               2                  anandh             anandhi@gmail.com                     anandhapriya                anandhi                   7865435643

16                    2015-11-12            720.00                   6                 madhu              madhu@gmail.com                     madhu                         madhu                     7865435643

17                    2015-12-12            500.00                 11                 arun                 arun@gmail.com                         arun                            aaa                         9078563454

18                   2015-07-12            920.00                   6                madhu             madhu@gmail.com                        madhu                      madhu                     7865435643

Enter the Transaction ID

**1**

Enter User Id

**3**

PurchaseId        Quantity            Cost

6               200.00               800.00

Enter Purchase Id

**6**

Complaint Types

1. Product Mismatch

2. Payment Related

3. Inappropriate Quantity

4. Order Delivery

5. Product Validity

Enter Complaint type

**1**

Enter Complaint Description

Product did not deliver properly

ComplaintId Complaint Type Id       Complaint Type Name                 Complaint Text

6                    1                    Product Mismatch                     Payment pending

11                    1                    Product Mismatch    Product did not deliver properly

**-------------------------**