**QUESTIONS(Any 8 questions)**

**Q-1:** Define a class called Account that holds account number, name of the person and balance. This class should have mechanism to perform deposit, withdraw and display the current balance. If the withdraw amount is more than balance then it must display "insufficient funds" otherwise the transaction will occur. Create at least one object of Account class in ATM class and display the final balance.

**Q-2:** Define a class called Author which will keep information like Author name, age, gender. Define a class called Book in which will have data of the book name, author(Reference variable of Author class), pages and cost, book number.

Define a class called Library in which create array of 5 Books and ask the user to choose book array index to get information of Book with its author.

**Q-3 :** Create an abstract class named Animal which includes abstract methods like eat() and sleep().

Create a child class of Animal named Bird and override the parent class methods. Add a new method named fly().

Create a child class of Animal named WildAnimal and override the parent class methods. Add a new method named kill().

Create an instance of Bird class and invoke the eat, sleep and fly methods using this object.

Create an instance of WildAnimal class and invoke the eat, sleep and kill methods using this object.

Create a reference of Animal class and invoke the eat and sleep methods using the object of Bird and WildAnimal.

**Q-4 :** Define a class called Candidate consisting of name, age, education. If age is less than 21 or greater than 60, then it will generate AgeException with information like ‘under age’ or ‘over age’ respectively. Similarly if the education if not Btech then it’ll generate EduException with a message ‘sorry, invalid education’. The AgeException and EduExceptions are user defined exception classes. Create object’s of Candidate class in main() method to validate the candidate’s age and education.

**Q-5**:A Hotel wants an application that fetches the details of the guests who have been staying there. They already have a database that has the details. It includes two modules:

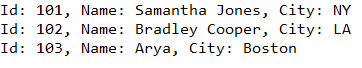
* Get All Guest Details
* Get All Guests By Name

**Method description**:

**getAllGuests()**:

* This method displays all the details of all the guests present in the database.
* Create the connection with the database by declaring all the required credentials and then by using the getConnection() method.
* Create the query to display the details of all the guests in the database
* Print the details
* Handle any exception that can occur
* Close the connection.

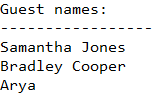
When this method is called, the expected output should be:



**getGuestsByName()**:

* This method displays the names of all the guests present in the database.
* Create the connection with the database by declaring all the required credentials and then by using the getConnection() method.
* Create the query to display the names of all the guests in the database
* Print the details
* Handle any exception that can occur
* Close the connection.

When this method is called, the expected output should be:



**Q-6:**  Develop a java class with a instance variable TreeMap. add a method listingCountry(String CountryName, ArrayList district) , the method should add the passed countryName as a key to TreeMap and ArrayList that contains the district names as a value of key.

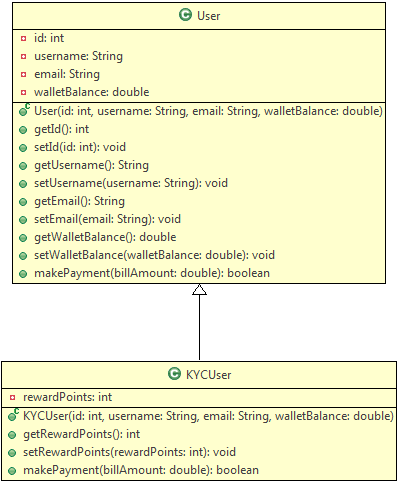
Develop a method getCountry(String CountryName) which iterates through the TreeSet and returns the ArrayList that has district name, but if CountryName does not exist then returns null.

Develop a method listCountryBefore(String CountryName) it must display all Country with their district before CountryName argument.

Develop a method listCountryAfter(String CountryName) it must display all Country with their district after CountryName argument.

Q:7 - **EPay Wallet is a wallet application using which its users can pay various bills. Users can make payments only if they have enough wallet balance.**

There are two kinds of users as illustrated by the class diagrams below. General users can make regular payments, whereas KYC users get reward points for every payment.



**Method Description:**

**User:**

**makePayment(double billAmount):** This method makes payment by deducting the bill amount from wallet if there is enough balance.

* If the balance is not enough, it returns false
* If the balance is sufficient, it deducts the specified bill amount from the wallet and returns true

**KYCUser:**

**makePayment(double billAmount):** This method overrides the parent method to make payment as well as to credit reward points to the user.

* It uses the payment functionality of the parent class
* If payment is successful, it adds 10% of the bill amount as reward points
* It returns true or false depending on whether the payment was successful or not

The primary EPay Wallet operations (currently only one) are to be defined in a separate class as follows:

This image shows the class diagram for class EPayWallet. It has one static method processPaymentByUser(user: User, billAmount: double): void

**Method Description:**

**processPaymentByUser(User user, double billAmount):** This is a static method to process the bill payment by any EPay Wallet user.

* It uses the makePayment() method of the user to process payments, and displays success or error messages depending on whether the payment was successful or not.
* It shows the wallet balance of the user
* If the user is a KYC user, it shows the reward points as well

**Note:**

* Have a look at the sample output to understand the messages to be displayed
* You can use Java's instanceOf operator to check the type of an object

Use the Tester class to test the above functionalities. Create User and KYCUser objects with different values and call the processPaymentByUser() method of EPayWallet class to process payments.

**Sample Input:​​​​​​​**

|  |  |
| --- | --- |
| **User** | |
| **id** | **101** |
| **username** | **Jack** |
| **email** | **jack@infy.com** |
| **walletBalance** | **1000** |

|  |  |
| --- | --- |
| **KYC User** | |
| **id** | **201** |
| **username** | **Jill** |
| **email** | **jill@infy.com** |
| **walletBalance** | **3000** |

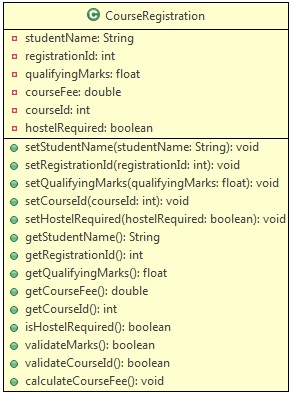
|  |  |
| --- | --- |
| **Sample payments** | |
| **Jack** | **700** |
| **Jill** | **1500** |
| **Jill** | **800** |
| **Jill** | **1200** |

Sample Output:

1. Congratulations Jack, payment of 700.0 was successful
2. Your wallet balance is 300.0
3. --------------------------------------------
4. Congratulations Jill, payment of 1500.0 was successful
5. Your wallet balance is 1500.0
6. You have 150 reward points
7. --------------------------------------------
8. Congratulations Jill, payment of 800.0 was successful
9. Your wallet balance is 700.0
10. You have 230 reward points
11. --------------------------------------------
12. Sorry Jill, not enough balance to make payment
13. Your wallet balance is 700.0
14. You have 230 reward points
15. --------------------------------------------

Q-8: An University wants to automate the process of registering students to different courses. The CourseRegistration class has been created to help in the process of course registration. The class diagram is given below:

**CourseRegistration:**



A student is eligible to get admission only if the qualifying marks is in the range of 65 to 100(both inclusive). At present, the university offers five courses with course ids in the range 1001 to 1005(both inclusive). Based on the marks in the qualifying exam, a discount is provided in the course fee. Use the table below to determine the discount:

| **Discount** | |
| --- | --- |
| **Marks Range** | **Discount** |
| **65-69** | **5%** |
| **70-84** | **10%** |
| **>=85** | **15%** |

Use the table below to get the base fee of different courses:

| **Base fee** | |
| --- | --- |
| **Course Id** | **Fee** |
| **1001** | **Rs.55,000** |
| **1002** | **Rs.35,675** |
| **1003** | **Rs.28,300** |
| **1004** | **Rs.22,350** |
| **1005** | **Rs.1,15,000** |

**Method Description:**

* **validateMarks()**: Used to validate qualifying exam marks - qualifying marks is in the range of 65 to 100(both inclusive)
* **validateCourseId()**: Used to validate the course entered, based on the courseId - given in the table above
* **calculateCourseFee()**: Used to calculate the course fee after applying the discount.

Use the Tester class to set the values and call appropriate methods to display the outputs for the inputs mentioned below:

**Sample:​​​​​​​**

| **Input 1** | |
| --- | --- |
| **Attributes** | **Values** |
| **Name** | **Peter** |
| **Registration Id** | **5001** |
| **Qualifying Exam Marks** | **58** |
| **Course Id** | **1005** |
| **Hostel Required** | **true** |

**Output:**

**Marks is less than 65. You are not eligible for admission!!**

| **Input 2** | |
| --- | --- |
| **Attributes** | **Values** |
| **Name** | **Peter** |
| **Registration Id** | **5001** |
| **Qualifying Exam Marks** | **68** |
| **Course Id** | **1006** |
| **Hostel Required** | **true** |

**Output:**

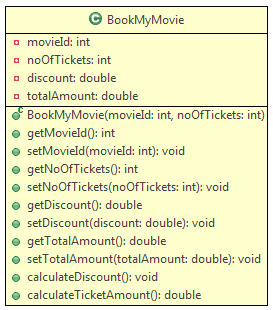
**Invalid Course Id. Please try again!!**​​​​​​​

| **Input 3** | |
| --- | --- |
| **Attributes** | **Values** |
| **Name** | **Peter** |
| **Registration Id** | **5001** |
| **Qualifying Exam Marks** | **78** |
| **Course Id** | **1005** |
| **Hostel Required** | **false** |

**Output:**

**\*\*\*\*\*\*\*\*Course Allocation Details\*\*\*\*\*\*\*\*  
                                                           Student Name                        :Peter  
                                                           Course Id                               :1005  
                                                           Qualifying Exam Marks        :78.0  
                                                           Student's Registration Id     :5001  
                                                           Total course fee                    :103500.0  
​​​​​​​                                                           Hostel Required                    : No**

Q-9: BookMyMovie is an online website, through which the required number of tickets can be booked for a particular show. A maximum number of 15 tickets can be issued per booking. Implement the class BookMyMovie as per the class diagram given below:



**Method Description:**

* **BookMyMovie(int movieId, int noOfTickets):** This constructor is used to initialize the instance variables with user provided values.
* **calculateDiscount():** This method is used to set the discount percentage for a particular booking based on the ***noOfTickets***issued and the ***movieId***. The condition for calculating the discount is given below:

| **Movie Data** | | |
| --- | --- | --- |
| **MovieId** | **noOfTickets** | **Value** |
| **101, 102, 103** | **<5** | **0** |
| **101 or 103** | **>=5 && <10** | **15** |
| **101 or 103** | **>=10 && <=15** | **20** |
| **102** | **>=5 && <10** | **10** |
| **102** | **>=10 && <=15** | **15** |

​​​​​​​

Note: In this above table, Value column stands for the discount percentage.

* **calculateTicketAmount():** This method is used to calculate the ticket amount after deducting the discount amount. It should call the method ***calculateDiscount()*** which sets the discount percentage. Then the ***totalAmount***should be calculated using the formula given below:
  + totalAmount = baseFare \* noOfTickets – (baseFare \* noOftickets \* (discount/100))

The baseFare for a given movieId is given below:

| **Movie Data** | |
| --- | --- |
| **MovieId** | **Base Fare** |
| **101** | **120** |
| **102** | **170** |
| **103** | **150** |

Once you implement the above class,

* Create a Tester class.
* Create objects of BookMyMovie class with the sample inputs given below.
* Invoke the **calculateTicketAmount()** for each of the object and verify with the sample output provided.

**Sample I/O:**

* **Input**(For BookMyMovie object 1):
  + *movieId*= 101
  + *noOfTickets*= 5
* **Output**(For BookMyMovie object 1): Total amount for booking: 510.0

* **Input**(For BookMyMovie object 2):
  + *movieId* = 102
  + *noOfTickets*= 4
* **Output**(For BookMyMovie object 2): Total amount for booking: 680.0

* **Input**(For BookMyMovie object 3):
  + *movieId*= 103
  + *noOfTickets* = 8
* **Output**(For BookMyMovie object3): Total amount for booking: 1020.0

* For any invalid *movieId*, the output should be,

1. Sorry! Invalid Movie ID!
2. Please check the Movie ID and enter once again.