Coding Activity: Hotel Billing System

LuxeStay is a luxury hotel chain that wants to automate its billing system for different types of rooms. You are building a **Hotel Billing System** that allows users to enter room details, calculate the total bill based on the number of nights stayed, and check if the stay qualifies for a loyalty discount based on the number of years since joining their membership program.

Component Specification: Room Interface

Type(Interface)	Methods	Responsibilities
Room	public double **calculateTotalBill**(int nightsStayed, int joiningYear)	This method should calculate and return the total bill for a room based on the number of nights stayed. It should be an abstract method and give implementation inside the HotelRoom class.
Room	public default int **calculateMembershipYears**(int joiningYear)	This method should calculates and returns the number of years since joining as a member (using the current year).

Component Specification: HotelRoom class, it should implement the Room interface

Component Specification: HotelRoom class, it should implement the Room interface

Type(Class)	Attributes	Responsibilities
HotelRoom	String roomType double ratePerNight String guestName	Appropriate setters, getters and three argument constructor are provided as a part of code skeleton.

Component Specification: HotelRoom class

Type(Class)	Methods	Responsibilities
HotelRoom	public double **calculateTotalBill**(int nightsStayed, int joiningYear)	• This method calculates and returns the total bill for a guest based on the number of nights stayed. • The total bill is computed using the formula: **TotalBill** = nightsStayed × ratePerNight. • If the guest has been a loyalty member for more than 3 years, a 10% discount is applied to the total bill (i.e., the guest pays 90% of the original amount). • Finally, the total bill is rounded to the nearest whole number before being returned.

In the UserInterface class, the prompt the user to enter details for both room types — Deluxe Room and Suite Room — following the format shown in

		• Finally, the total bill is rounded to the nearest whole number before being returned.

In the **UserInterface** class, the prompt the user to enter details for both room types — Deluxe Room and Suite Room — following the format shown in the sample input and output. Initialize the values to **HotelRoom** class based on the room type.(Deluxe or Suite).

Based on the entered details, create **HotelRoom** objects and initialize the room type, rate per night, and guest name.

Next, invoke the default method **calculateMembershipYears()** by passing the guest's joining year to determine their total years of membership.

After that, invoke the method **calculateTotalBill()** by passing the number of nights stayed and joining year to calculate the final bill amount, applying discounts when applicable.

Finally, display a summary of each guest's room details, membership duration, and the calculated total bill in the specified output format.

Note:

- **roomType** is case-sensitive.
- In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user, and the rest of the text represents the output.
- Ensure to follow the object-oriented specifications provided in the question description.
- Ensure to provide the names for classes, attributes, and methods as specified in the question description.
- Adhere to the code template, if provided.
- **Please do not use System.exit(0) to terminate the program.**

Sample Input and Output 1:

Sample Input and Output 1:

Enter Deluxe Room Details:

Guest Name: **Suresh**

Rate per Night: **8999**

Nights Stayed: **3**

Joining Year: **2019**

Enter Suite Room Details:

Guest Name: **Sunita**

Rate per Night: **9999**

Nights Stayed: **4**

Joining Year: **2024**

Room Summary:

Deluxe Room: Suresh, 8999.0 per night, Membership: 6 years

Suite Room: Sunita, 9999.0 per night, Membership: 1 years

Enter Suite Room Details:

Guest Name: Sunita

Rate per Night: 9999

Nights Stayed: 4

Joining Year: 2024

Room Summary:

Deluxe Room: Suresh, 8999.0 per night, Membership: 6 years

Suite Room: Sunita, 9999.0 per night, Membership: 1 years

Total Bill:

For Suresh (Deluxe): 24297.0

For Sunita (Suite): 39996.0

Coding Activity: XamXpert

Description | Submission | Code Editor | Grading view

XamXpert is an online examination platform that wants to automate its test evaluation process for different types of questions. You are building a Test Evaluation System that allows users to enter exam details, calculate the total score percentage based on correct and wrong answers, and determine the final result (Merit, Pass, or Fail) using a static method defined in an interface.

Component Specification: Exam Interface

Type	Methods	Responsibilities
Interface	public double **calculateScore**()	This is an abstract method to calculate and return the score percentage for a student based on the question type , number of correct and wrong answers. The implementation is provided inside the OnlineTest class.
Interface	public static String **evaluateResult**(double percentage)	This is a static method that evaluates and returns the final result (Merit, Pass, or Fail) based on the percentage score. If the percentage is greater than or equal to 85, the student is classified as "Merit".

XamXpert is an online examination platform that wants to automate its test evaluation process for different types of questions. You are building a Test Evaluation System that allows users to enter exam details, calculate the total score percentage based on correct and wrong answers, and determine the final result (Merit, Pass, or Fail) using a static method defined in an interface.

Component Specification: Exam Interface

Type	Methods	Responsibilities
Interface	public double **calculateScore()**	This is an abstract method to calculate and return the score percentage for a student based on the question type , number of correct and wrong answers. The implementation is provided inside the OnlineTest class.
Interface	public stati String **evaluateResult**(double percentage)	This is a static method that evaluates and returns the final result (Merit, Pass, or Fail) based on the percentage score. If the percentage is greater than or equal to 85, the student is classified as "Merit". if percentage is greater than or equal to 60 but less than 85, the result is "Pass". otherwise, the student is considered to be "Fail".

Component Specification: OnlineTest Class, it should implement the Exam interface

Type	Attributes	Responsibilities
	String studentName int totalQuestions	Appropriate setters, getters and five

Component Specification: OnlineTest Class, it should implement the Exam interface

Type	Attributes	Responsibilities
OnlineTest	String studentName int totalQuestions int correctAnswers int wrongAnswers String questionType	Appropriate setters, getters and five argument constructor is provided as a part of code skeleton

Component Specification: OnlineTest class

Component Name	Type(Class)	Methods	Responsibilities
Returning total score percentage	OnlineTest	public double calculateScore()	This method calculates and returns the percentage score of a student based on the question type and the number of correct and wrong answers. • For MCQ type → each correct answer carries 2 marks. • For Coding type → each correct answer carries 5 marks. • For every wrong answer, 10% of marks per question are deducted as a penalty. Formula: totalScore=(correctAnswers * marksPerQuestion)-(wrongAnswers *

Component Specification: OnlineTest class

Component Name	Type(Class)	Methods	Responsibilities
Returning total score percentage	OnlineTest	public double calculateScore()	This method calculates and returns the percentage score of a student based on the question type and the number of correct and wrong answers. • For MCQ type → each correct answer carries 2 marks. • For Coding type → each correct answer carries 5 marks. • For every wrong answer, 10% of marks per question are deducted as a penalty. **Formula:** totalScore=(correctAnswers * marksPerQuestion)-(wrongAnswers * (marksPerQuestion*0.10)) percentage=(totalScore /(totalQuestions * marksPerQuestion))×100

Example:
If there are 10 coding questions, out of which 7 are correct and 3 are wrong:

totalScore=(7*5)-(3*(5*0.10))=35-1.5=33.5

percentage=(33.5/((10*5)))*100=67%

Hence, the student scored 67% in the test.

In the UserInterface class, prompt the user to enter the exam details such as student name, question type (MCQ or Coding), total number of

Component Specification: OnlineTest class

Component Name	Type(Class)	Methods	Responsibilities
Returning total score percentage	OnlineTest	public double calculateScore()	This method calculates and returns the percentage score of a student based on the question type and the number of correct and wrong answers. • For MCQ type → each correct answer carries 2 marks. • For Coding type → each correct answer carries 5 marks. • For every wrong answer, 10% of marks per question are deducted as a penalty. **Formula:** totalScore=(correctAnswers * marksPerQuestion)-(wrongAnswers * (marksPerQuestion*0.10)) percentage=(totalScore /(totalQuestions * marksPerQuestion))×100

Example:
If there are 10 coding questions, out of which 7 are correct and 3 are wrong:

totalScore=(7*5)-(3*(5*0.10))=35-1.5=33.5

percentage=(33.5/((10*5)))*100=67%

Hence, the student scored 67% in the test.

In the UserInterface class, prompt the user to enter the exam details such as student name, question type (MCQ or Coding), total number of questions, number of correct answers, and number of wrong answers.

Example:

If there are 10 coding questions, out of which 7 are correct and 3 are wrong:

totalScore=(7*5)-(3*(5*0.10))=35-1.5=33.5

percentage=(33.5/((10*5)))*100=67%

Hence, the student scored 67% in the test.

In the **UserInterface** class, prompt the user to enter the **exam details** such as student name, question type (MCQ or Coding), total number of questions, number of correct answers, and number of wrong answers.

Then initializes the values into an **OnlineTest** object and invoke the **calculateScore()** method to compute the percentage score.

Next, invoke the **static method evaluateResult()** to determine whether the student has achieved **Merit**, **Pass**, or **Fail** based on the percentage obtained.

Finally, display the result as shown in the sample output.

Note:

- **questionType** is case-sensitive.
- In the output the **totalScore** should be displayed with exactly **one decimal place** as shown in sample output.
- In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user, and the rest of the text represents the output.
- Ensure to follow the object-oriented specifications provided in the question description.
- Ensure to provide the names for classes, attributes, and methods as specified in the question description.
- Adhere to the code template, if provided.
- **Please do not use System.exit(0) to terminate the program.**

Sample Input 1

Enter Exam Details:

Student Name:

Sara Khan

Question Type (MCQ/Coding):

Coding

Total Questions:

20

Correct Answers:

18

Wrong Answers:

2

Sample Output 1

Exam Summary:

Coding Test: Sara Khan, Total Score: 89.0, Result: Merit

Sample Input 2

Enter Exam Details:

Student Name:

Rio

Question Type (MCQ/Coding):

MCQ

Total Questions:

25

Correct Answers:

19

Wrong Answers:

6

Sample Output 2

Exam Summary:

MCQ Test: Rio, Total Score: 73.6, Result: Pass