1.Vehicle Fleet Compilance

RapidTrans Logistics is dedicated to ensuring the compliance and safety of its vehicle fleet. To support this initiative, they are developing a robust system to validate vehicle registration details through their online portal. This validation system will help ensure that all vehicles meet regulatory standards, reduce the risk of non-compliance fines, and enhance overall fleet safety.

Your task is to design a Java program to obtain vehicle details (registration ID, vehicle license number, driver license number) and validate this information, incorporating exception handling to manage various error scenarios.

Component Specification: VehicleRegistrationSystem

Type (Class) Method Responsibilities

VehicleRegistrationSystem validateFleetDetails This method takes three arguments: registrationId as a string, vehicleLicenseNumber as a string, and driverLicenseNumber as a string. It validates the input as per the validation rules. If all details are valid, return a string "Details for Registration ID:<registrationId> are valid and can be added to the system" else throw an InvalidVehicleDetailsException.

Component Specification: InvalidVehicleDetailsException (This class inherits the Exception Class)

Type (Class) Responsibilities

InvalidVehicleDetailsException Provided with a single-argument constructor: public InvalidVehicleDetailsException (String message). The exception is thrown when the registrationId, vehicleLicenseNumber, or driverLicenseNumber does not follow the below validation rules.

Validation Rules:

1. The registration ID should match the pattern "VR-XXXX", where "XXXX" is a 4-digit number. [e.g.: VR-4519]

2. The vehicle license number should match the pattern "AAXXAAXXXX", where the first two characters are uppercase alphabets, the next two are digits, the next two are uppercase alphabets, and the last four are digits. [eg:TN63DE1366]

3. The driver license number should match the pattern "AAXXXXXXXXXXXXX", where the first two characters are uppercase alphabets, and the next thirteen characters are digits.

Propagate the exceptions that occur in the VehicleRegistrationSystem class and handle them in the main method.

If the registrationId is invalid, an exception should be raised with the message "Invalid Registration ID".

If the vehicleLicenseNumber is invalid, an exception should be raised with the message "Invalid Vehicle License Number".

If the driverLicenseNumber is invalid, an exception should be raised with the message "Invalid Driver License Number".

Note:

- In the sample input and 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 don't use System.exit(0) to terminate the program.

Sample Input / Output 1:

Enter Registration ID:

VR-5118

Enter Vehicle License Number:

TN52GF7239

Enter Driver License Number:

DL2331509347998

Details for Registration ID: VR-5118 are valid and can be added to the system

Sample Input / Output 2:

Enter Registration ID:

vr-5117

Enter Vehicle License Number:

KL23HL1328

Enter Driver License Number:

VR9811200596225

Invalid Registration ID

Sample Input / Output 3:

Enter Registration ID:

VR-8112

Enter Vehicle License Number:

KL126725

Enter Driver License Number:

VS7801620553711

Invalid Vehicle License Number

Sample Input / Output 4:

Enter Registration ID:

VR-6299

Enter Vehicle License Number:

TN55SL3555

Enter Driver License Number:

DL213378

Invalid Driver License Number

2.Carrier Link

Carrier Link, a leading recruitment consultancy firm, is developing a Recruitment Consultancy System to streamline and automate its recruitment process. The system will validate applicant details and calculate consultancy fees based on user inputs, ensuring accuracy and efficiency. It will handle applicant information and incorporate exception handling for various error scenarios effectively.

Your task is to design a Java program to obtain applicant details (applicantId, position applied for, reference code, and years of experience), validate this information, and calculate the appropriate consultancy fee, providing clear error messages for invalid inputs.

Component Specification: RecruitmentConsultancySystem

Type (Class)	Method	Responsibilities
RecruitmentConsultancySystem	validateApplicantDetails	This method validates the applicant's details according to the specified validation rules. If all details are valid, it returns true. If any of the details are invalid, it throws an InvalidApplicantException with the appropriate error message.
RecruitmentConsultancySystem	calculateConsultancyFee	This method calculates and returns the consultancy fee based on the position and yearsOfExperience. consultancyFee = baseFee * experienceMultiplier

Component Specification: InvalidApplicantException

Type (Class)	Responsibilities
InvalidApplicantException	Provided with a single-argument constructor: public InvalidApplicantException(String message). It is thrown when the applicantId, position, or referenceCode does not follow the validation rules..

Validation Rules:

1. applicantId: Must be exactly six characters. The first two characters must be uppercase alphabets followed by four digits.[e.g. QK4863]
2. position: Must be either "Junior", "MidLevel", or "Senior" (case-sensitive).
3. referenceCode: Must start with any of the three uppercase alphabet 'J', 'M' or 'S' which indicates the position (J for Junior, S for Senior, M for MidLevel), followed by four digits representing the year of application, then a hypen (-) , followed by 3 digits.[e.g. J2024-173]

Guidelines to calculate the consultancy fee based on the position:

position	baseFee
Junior	500
MidLevel	1000
Senior	1500

Experience Multiplier:

Years of Experience	experienceMultiplier
> 2 and < 5	1.1
> 5 and < 10	1.3
> 10	1.5

Explanation:

Let's consider an example where the applicant has applied for a "MidLevel" position and has 7 years of experience.

The baseFee for the "MidLevel" position is 1000.

And the yearsOfExperience 7, falls between 5 and 10 years, so the experienceMultiplier is 1.3

consultancyFee = baseFee * experienceMultiplier

= 1000 * 1.3

= 1300

Component Specification: UserInterface

Note:

- Propagate the exceptions that occur in the RecruitmentConsultancySystem class and handle them in the main method.
- If the applicantId is invalid, display the message "Invalid applicant ID".
- If the position is invalid, display the message "Invalid position".
- If the referenceCode is invalid, display the message "Invalid reference code".
- Assume that the yearsOfExperience is always valid.
- In the sample input and 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 don't use System.exit(0) to terminate the program.

Sample Input / Output 1:

Enter the applicant ID

WR3619

Enter the position applied for

MidLevel

Enter the reference code

M2023-812

Enter the years of experience

5

Validation Successful

Consultancy Fee: 1100.0

Sample Input / Output 2:

Enter the applicant ID

ag3411

Enter the position applied for

Junior

Enter the reference code

J2024-562

Enter the years of experience

3

Invalid applicant ID

Sample Input / Output 3:

Enter the applicant ID

AH6700

Enter the position applied for

senior

Enter the reference code

S2021-115

Enter the years of experience

10

Invalid position

Sample Input / Output 4:

Enter the applicant ID

PW5779

Enter the position applied for

Junior

Enter the reference code

L2024-18

Enter the years of experience

3

Invalid reference code

Malar Senior Homage

Malar Senior Homage founded by philanthropist Malar Ramesh to offer free religious tours across the country for senior citizens. They collect and verify applicants details like name, age, phone number and preferred travel mode. Once validated, applicants receive a unique ID to join the tours. The program aims to fulfil seniors' spiritual dreams and create a supportive community, with each journey accompanied by volunteers and medical staff for a safe and enriching experience.

You as a software developer help Malar Ramesh in creating an application to meet her requirements.

Consider the following scenario:

You receive input that is the name as string, age as int, phoneNumber as long, and travelMode in the format of a string in the UserInterface class.

Component Specification: Class - TouristManagementSystem

Method	Parameters	Responsibilities
validateTouristDetails	String name, int age, long phoneNumber, String travelMode	This method takes four arguments name as String, age as int, phoneNumber as long, travelMode as String and validate the same as per the validation rules.

Validation Rules:

1. If the name contains other than uppercase and lowercase alphabets and spaces, then InvalidTouristDetailsException is raised with the message as "<name> is an invalid name".

2. If the age is less than 60 or greater than 99, then InvalidTouristDetailsException is raised with the message as "<age> is an invalid age".

3. If the phoneNumber is not exactly 10 digits, then InvalidTouristDetailsException is raised with the message as "<phoneNumber> is an invalid phone number".

4. The travelMode can be either one of [Airway, Roadway], if not then InvalidTouristDetailsException is raised with the message as "<travelMode> is an invalid travel mode".

If valid return true else throw "InvalidTouristDetailsException"

getTouristId	String name, int age, long phoneNumber, String travelMode	This method generates a tourist ID using four arguments: name (a String), age (an int), phoneNumber (a long), and travelMode (a String). The ID is created by taking the first two characters from each of these inputs. This method returns a message "Tourist details are valid your tourist Id is <generated touristId>" if all the details are valid.

Example: If name is Sakthi, age is 75, phoneNumber is 8978967565, and travelMode is Airways, then the tourist Id generated should be Sa7589Ai.

Condition: The tourist Id generated is case-sensitive and should be exactly 8 characters.

Component Specification: InvalidTouristDetailsException (This class inherits the Exception Class)

Type (Class)	Responsibilities
InvalidTouristDetailsException	Provided with a single-argument constructor: public InvalidTouristDetailsException (String message). Thrown when name, age, phoneNumber, TravelMode does not follow the below validation rules.

Component Specification: UserInterface

Note:

- Propagate the exceptions that occur in TouristManagementSystem class and handle them in main method.
- In the sample input and 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 / OUTPUT 1:

Enter your name

John

Enter your age

80

Enter your phone number

6798670987

Enter your travel mode

Airway

Tourist details are valid your tourist Id is Jo8067Ai

SAMPLE INPUT / OUTPUT 2:

Enter your name

Jenny

Enter your age

54

Enter your phone number

8767890876

Enter your travel mode

Airway

54 is an invalid age

SAMPLE INPUT / OUTPUT 3:

Enter your name

Joseph

Enter your age

78

Enter your phone number

8790678

Enter your travel mode

Roadway

8790678 is an invalid phone number

SAMPLE INPUT / OUTPUT 4:

Enter your name

Sakthi

Enter your age

90

Enter your phone number

7890678567

Enter your travel mode

Waterway

Waterway is an invalid travel mode

SAMPLE INPUT / OUTPUT 5:

Enter your name

1@Sakthi

Enter your age

90

Enter your phone number

7890678567

Enter your travel mode

Airway

1@Sakthi is an invalid name

3.Recurring Savings Scheme

ZAB Bank introduces the "Recurring Savings Scheme" and requires an application to validate customer records. The records include account number, name, age, tenure, and monthly saving amount. The application ensures each detail meets specific validation conditions. If any detail is invalid, the application handles exceptions with appropriate messages. Upon successful validation, it calculates and displays the total interest earned along with the principal amount. As a software developer, help ZAB Bank achieve this task.

Component Specification: Class - CustomerInfoSystem

Method Parameters Responsibilities

validateCustomerInfo String accountNumber, String name, int age, int tenure, double monthlyDeposit This method takes five arguments accountNumber as String, name as String, age as int, tenure as int, monthlyDeposit as double and validate the same as per the validation rules.

Validation Rules:

1. If accountNumber is not a 10 digit number, then an InvalidCustomerInfoException is raised with the message "<accountNumber> is an invalid account number".

2. If the name contains other than uppercase and lowercase alphabets and spaces, then an InvalidCustomerInfoException is raised with the message "<name> is an invalid name".

3. If the age is not between the inclusive range of 18 to 100, then an InvalidCustomerInfoException is raised with the message "<age> is an invalid age".

4. If the tenure is not between the inclusive range of 12 to 120 months, then an InvalidCustomerInfoException is raised with the message "<tenure> is an invalid tenure".

5. The monthlyDeposit should be a positive number, if not then an InvalidCustomerInfoException is raised with the message "<monthlyDeposit> is an invalid monthly deposit".

If all details are valid the method returns true else, throws an InvalidCustomerInfoException.

getTotalAmount int age, int tenure, double monthlyDeposit This method takes age, tenure and monthlyDeposit as parameters and determine the interest based on age and then calculate the interest earned and sum up with the principal amount and return total amount as double.

age (inclusive ranges) monthly rate of interest

18-59 2%

60-100 5%

Example: If age=45, monthlyDeposit=5000, tenure=15 months then monthly rate of interest will be 2%

Monthly Interest amount = (5000*0.02) = 100

For 15 month, Total interest is 15*100 = 1500

Total Amount = principal amount + total interest = 5000*1500 = 6500

Note: Use 0.02 directly for the interest calculation instead of 2/100.[because 2/100 results in 0 due to integer division].

Component Specification: InvalidCustomerInfoException (This class inherits the Exception Class)

Type (Class) Responsibilities

InvalidCustomerInfoException Provided with a single-argument constructor: public InvalidCustomerInfoException (String message). Thrown when accountNumber, name, age, tenure, monthlyDeposit does not follow the validation rules.

Component Specification: UserInterface

Type (Class) Responsibilities

UserInterface In the main method get accountNumber, name, age, tenure, monthlyDeposit from the user. Invoke the validateCustomerInfo method to validate these details. If all the details are valid invoke the getTotalAmount method to calculate the totalAmount. Handle any InvalidCustomerInfoException that are thrown and display the appropriate error message to the user.

Note:

- Propagate the exceptions that occur in CustomerInfoSystem class and handle them in main method.
- In the sample input and 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:

Enter the account number

5878903457

Enter the name

John

Enter the age

55

Enter the tenure

24

Enter the monthly deposit

3000

Total amount is 4440.0

SAMPLE INPUT AND OUTPUT 2:

Enter the account number

12345678901

Enter the name

Hannah

Enter the age

45

Enter the tenure

57

Enter the monthly deposit

4500

12345678901 is an invalid account number

SAMPLE INPUT AND OUTPUT 3:

Enter the account number

6746892348

Enter the name

Joan

Enter the age

12

Enter the tenure

78

Enter the monthly deposit

9000

12 is an invalid age

SAMPLE INPUT AND OUTPUT 4:

Enter the account number

4567876789

Enter the name

Merin

Enter the age

46

Enter the tenure

10

Enter the monthly deposit

8000

10 is an invalid tenure

SAMPLE INPUT AND OUTPUT 5:

Enter the account number

6789765678

Enter the name

Jonny

Enter the age

89

Enter the tenure

30

Enter the monthly deposit

-9

-9.0 is an invalid monthly deposit

4.Security Key Generation

To enhance security against potential attacks, a famous online market requires customers to provide a Security Key derived from their username before making a purchase. This Java application is designed to generate the Security Key based on specific criteria.

Note:

1. The length of the string should be between 2 and 10, both inclusive. Else print "Invalid Input" and terminate the program.

2. Input must consist of a single string (upper case or lower case) which corresponds to the username of the user, without any spaces.

3. The username must consist only of alphabetic characters (both uppercase and lowercase). Else print "Invalid Username" and terminate the program.

For generating the Security Key:

1. Frequency Calculation: Convert the username to lowercase and calculate the frequency of each character. Construct the first part of the Security Key by appending each unique character followed by its frequency.

2. Average ASCII Character: Convert the original username to lowercase, calculate the sum of ASCII values for all characters, then compute the average. Convert this average to a character and append it to the end of the Security Key.

Note:

- 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.

- Adhere to the code template if provided.
- Do not use System.exit(0) to terminate the code.

Sample Input / Output 1

Enter the username:

Reverse

Security Key: r2e3v1s1m

Explanation:

1. Frequency Calculation: The frequency of each character in the lowercase version of "Reverse" is calculated, resulting in "r2e3v1s1" for the Security Key.

2. ASCII Calculation: The sum of ASCII values for "reverse" is 764, with an average of 109. The character corresponding to this average, 'm', is appended to the Security Key. Thus, the final Security Key is"r2e3v1s1m".

Sample Input / Output 2

Enter the username:

A

Invalid Input

Sample Input / Output 3

Enter the username:

ab@cd

Invalid Username

Pothys Word Battle

Pothys has announced an exciting offer with a 50% discount. To avail of this offer, customers need to complete a specific task involving counting letters in a sentence. Here's how it works:

Task: Customers will receive a sentence and need to analyze each word based on the number of letters it contains.

Rules:

- For each word:Count the total number of letters, and determine whether the count is even or odd.
- If the letter count is even: Count the unique vowel letters (a, e, i, o, u) in that word.
- If the letter count is odd: Count the unique consonant letters in that word.
- For each word, print the results based on the counts:
- If a word contains no vowels or consonants, the program should print: "There are no vowels/consonants in the word <word>".
- If a word contains exactly one vowel or consonant, the program should print: "There is 1 vowel/consonant in the word <word>".
- If a word contains more than one vowel or consonant, the program should print: "There are <count> vowels/consonants in the word '<word>'".

- The input sentence should contain only alphabets and spaces. If it includes any other characters, the program should print: "Enter a valid sentence, the sentence should contain only alphabets and spaces."

Example:

Let us take an example sentence , say, "Coding is fun"

Word 1: "Coding"

Letter Count: 6 (even number)

Action: Count the unique vowels.

Unique Vowels: o, i

Count: 2

Output: "There are 2 vowels in the word Coding"

Word 2: "is"

Letter Count: 2 (even number)

Action: Count the unique vowels.

Unique Vowels: i

Count: 1

Output: "There is 1 vowel in the word is"

Word 3: "fun"

Letter Count: 3 (odd number)

Action: Count the unique consonants.

Unique Consonants: f, n

Count: 2

Output: "There are 2 consonants in the word fun"

Note:

- 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.
- Adhere to the code template, if provided.
- Do not use System.exit(0) to terminate the code.

Sample input 1:

Enter the sentence

Fortune favors the bold

Sample output 1:

There are 4 consonants in the word fortune

There are 2 vowels in the word favors

There are 2 consonants in the word the

There is 1 vowel in the word bold

Sample input 2:

Enter the sentence

I love Coimbatore

Sample output 2:

There are no consonants in the word i

There are 2 vowels in the word love

There are 4 vowels in the word coimbatore

Sample input 3:

Enter the sentence

Trust the process

Sample output 3:

There are 3 consonants in the word trust

There are 2 consonants in the word the

There are 4 consonants in the word process

Sample input 4:

Enter the sentence

He is a developer

Sample output 4:

There is 1 vowel in the word he

There is 1 vowel in the word is

There are no consonants in the word a

There are 5 consonants in the word developer

Sample input 5:

Enter the sentence

4km distance

Sample output 5:

Enter a valid sentence, the sentence should contain only alphabets and spaces

5.Character Count

Alex aims to develop a program that takes a word as input and counts the occurrences of each character within the word. He then plans to replace each letter with its respective count of occurrence in the word, transforming the word into a sequence of numbers that represent the frequency of its characters. This method will help Alex quickly identify and analyze the frequency of characters in any given string. Help Alex to perform this task.

Note:

- If the input word contains any special character or digit or space, print "<word> is an invalid word".
- The character count in a word should be calculated regardless of the case.
- Display output as given in the sample input 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.
- Adhere to the code template, if provided.
- Do not use System.exit(0) to terminate the code.

Sample Input / Output 1:

Enter a word

programming

12121211

{Explanation:

Letters	p	r	o	g	a	m	i	n
Occurrences	1	2	1	2	1	2	1	1

}

Sample Input / Output 2:

Enter a word

HapPiness

1121112

Sample Input / Output 3:

Enter a word

dee56 @

dee56 @ is an invalid word

6.Country Code

Nizaam Overseas Agency is developing a system to identify the country associated with a client's phone number based on the ZIP code. Initially, the system will support a limited range of countries but will be designed for future expansion. The system will use the following ZIP codes:

Country (case-sensitive)	India	Egypt	Australia	China	Japan	Singapore	UnitedKingdom	Afghanistan
Code	+91	+20	+61	+86	+81	+65	+44	+93

The system will also generate a secret phone number code by:

1. Splitting the phone number into 2-character segments.
2. Converting these segments into ASCII characters.
3. Printing the ASCII characters along with the corresponding country name.

Example: For the phone number +91-8978978675, the segments 89, 78, 97, 86, and 75 convert to ASCII characters Y, N, a, V, and K, identifying the country as India, with the output: "IndiaYNaVK".

Constraints:

- If the Code is invalid, such that not as per any one from the above table, then display "<phone number> is an unknown number" and terminate the program.
- When no number is entered,
- When the number doesn't start with (+),
- When the length of the number is not 14,
- When the 3rd character is not (-),
- When the string from the 4th index is not 10 characters long,

display "<phone number> is an invalid phone number" and terminate the program.

Note:

- 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.
- Please do not use System.exit(0) to terminate the program.

Sample Input / Output 1:

Enter the phone number :

+91-9384758399

India]TKSc

Sample Input / Output 2:

Enter the phone number :

+93-289282999

+93-289282999 is an invalid phone number

Sample Input / Output 3:

Enter the phone number :

+66-9876543210

+66-9876543210 is an unknown number

7.Pizza Hut

Pizza Hut is a famous Pizza shop across the city. They wanted to get the order details based on the pizza type. The manager intimates a software developer to help in their process. You are being the software developer, develop a Java program based on the requirement.

Component Specification: PizzaOrderInfo Class

Type (Class)	Attributes	Methods
PizzaOrderInfo	private Set<String> orderSet	Getter and setter methods for the attribute are included in the code skeleton.

Note: In the orderSet, each entry holds an orderId and pizzaType separated by a colon.

Requirement 1: Add the pizza order details to the orderSet.

Type (Class) Methods Responsibilities

PizzaOrderInfo public void addPizzaOrderDetails(String orderDetail) This method accepts orderDetail as an argument. It should add these details to the orderSet (TreeSet).

Requirement 2: Find the number of orders based on the given pizzaType

Type (Class) Methods Responsibilities

PizzaOrderInfo public int findTheNumberOfOrdersBasedOnThePizzaType(String pizzaType) This method accepts pizzaType as an argument. If the pizzaType matches the pizzaType present in the TreeSet, it should count the number of orders and return the same. Else return -1.

Condition: pizzaType is a case-sensitive

Requirement 2: Filter the Order details based on the pizzaType.

Type (Class) Methods Responsibilities

PizzaOrderInfo public List<String> findOrderDetailsBasedOnPizzaType(String pizzaType) This method filters the orders and returns the list of Order details with the same pizzaType .

Condition: pizzaType is a case-sensitive

In the main method of the UserInterface class, first obtain the number of orders from the user, then prompt for the order details.

Invoke the addPizzaOrderDetails method by passing the pizza details as colon separated and store them in the orderSet which is a TreeSet.

Get the pizzaType from the user and invoke the findTheNumberOfOrdersBasedOnThePizzaType method by passing the pizzaType to find the count of orders for the specified pizzaType. If no orders are available, print "No orders were found for <pizzaType>"; otherwise, print the count of orders as shown in the sample output.

Get the pizzaType from the user. Invoke the findOrderDetailsBasedOnPizzaType method by passing the pizzaType to filter and retrieves order details based on a specified pizzaType and add it to the list. If orders details are found, print the order details as shown in sample output. If no matching applications found, print "No Order details were found for the pizza - <pizzaType >".

Note:

- 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 the classes, attributes, and methods as specified in the question description.
- Adhere to the code template, if provided.
- Please do not use System.exit(x) to terminate the code.

Sample Input/Output 1:

Enter number of orders to be added

6

Enter the orders (Order Id: Pizza Type)

8028:ChilliPizza

3643:SicilianPizza

8293:PizzaFritta

1878:ChilliPizza

2938:GrillPizza

3643:SicilianPizza

Enter the Pizza type to be searched

SicilianPizza

The orders based on SicilianPizza are 1

Enter the Pizza type to identify the Order details

ChilliPizza

Orders based on the ChilliPizza are

Order Id: 1878, Pizza Type: ChilliPizza

Order Id: 8028, Pizza Type: ChilliPizza

Sample Input/Output 2:

Enter number of orders to be added

2

Enter the orders (Order Id: Pizza Type)

7656:ChilliPizza

2234:SicilianPizza

Enter the Pizza type to be searched

PizzaFritta

No orders were found for PizzaFritta

Enter the Pizza type to identify the Order details

GrillPizza

No Order details were found for the pizza - GrillPizza

8.Magic Touch

Magic Touch is a well-known AC Maintenance company across the city. They wanted to get the Service code based on the AC type. The manager intimates a software developer to help in their process. You, being the software developer, develop a Java program based on the requirement.

Component Specification: MaintenanceInfo Class

Type (Class) Attributes Methods

MaintenanceInfo Map<String,String> maintenanceMap Getter and setter methods for the attribute are included in the code skeleton.

Note: Here the maintenanceMap, holds the Key as serviceCode and Value as acType.

Requirement 1: Add the Ac Maintenance details to the maintenanceMap

Type (Class) Methods Responsibilities

MaintenanceInfo public void addAcMaintenanceDetails(String serviceCode, String acType) This method accepts serviceCode and acType as the arguments. It should add these details to the maintenanceMap (TreeMap).

Requirement 2: Find the number of records based on the given acType.

Type (Class) Methods Responsibilities

MaintenanceInfo public int findTheNumberOfRecordsBasedOnTheACType(String acType) This method accepts acType as an argument. If the acType matches the acType present in the Map, it must count the Ac details and return the same. Else return -1.

Condition: acType is a case-sensitive.

Requirement 3: Filter the Service codes based on the acType.

Type (Class)	Methods	Responsibilities
MaintenanceInfo	public Set<String> findServiceCodesBasedOnACType(String acType)	This method accepts acType as the input parameter and it filters the Ac details and returns the Set of Service codes which has the same acType. Condition: acType is a case-sensitive.

In the main method of the UserInterface class, obtaining the number of Ac maintenance details and the Ac maintenance details from the user is given as code skeleton.

Invoke the addAcMaintenanceDetails method by passing the serviceCode and acType parameters and store them in the maintenanceMap which is a TreeMap.

Get the acType from the user to find the count of the Ac maintenance details based on the specified acType and invoke the findTheNumberOfRecordsBasedOnTheACType method by passing the acType. If no details are available, print "No records were found for <acType >"; otherwise, print the count of details as shown in the sample output.

Get the acType from the user. Invoke the findServiceCodesBasedOnACType method by passing the acType parameter to filter and retrieves Ac maintenance details based on a specified acType and add it to the HashSet. If details are found, print the Ac Maintenance details as shown in sample output. If no details are available, print "No service codes were found for the type <acType>".

Note:

- 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 the classes, attributes, and methods as specified in the question description.
- Adhere to the code template, if provided.
- Please do not use System.exit(x) to terminate the code.

Sample Input/Output 1:

Enter number of records to be added

4

Enter the records (Service code: AC type)

CVW:Window

ASF:Split

AJS:Window

CVW:Window

Enter the AC type to be searched

Split

The records based on Split are 1

Enter the AC Type to identify the Service codes

Window

Records based on the Window are

CVW

AJS

Sample Input/Output 2:

Enter number of records to be added

2

Enter the records (Service code: AC type)

SJH:Window

JAS:Split

Enter the AC type to be searched

FloorMounted

No records were found for FloorMounted

Enter the AC Type to identify the Service codes

Smart

No service codes were found for the type Smart

9.Book Heaven

"Book Haven," a popular subscription service, needs a system to efficiently manage users. They require functionality to add new clients and look up client IDs by subscription packages. The manager has tasked you, a software developer, with creating a Java program to implement these features seamlessly.

Component Specification: ClientInfo Class

Type (Class)	Attributes	Methods
ClientInfo	private Set<String> clientSet	Getter and setter methods for the attribute are included in the code skeleton.

Note: Here, the clientSet holds clientId and packageName, separated by a colon (.

Requirement	Type (Class)	Methods	Responsibilities
Adds clientDetails(clientId :packageName) to clientSet.	ClientInfo	public void addClient(String clientDetails)	This method takes clientDetails as parameters. It inserts the provided clientDetails into the clientSet.
Find the count of client IDs based on the given packageName.	ClientInfo	public int getTotalClientsByPackageName(String packageName)	This method accepts packageName as an argument, iterates over each client in the Set, splits the string using the colon delimiter, and checks if the packageName matches the provided packageName. If a match is found, it counts the number of clients and returns the same. If no matches are found, return -1. Condition: packageName is case-sensitive
Filter the client Ids based on the packageName.	ClientInfo	public List<String> listClientIdsByPackageName(String packageName)	This method accepts packageName as an argument, iterates over each client in the Set, splits the string using the colon delimiter, and checks if the packageName matches the provided packageName. If a match is found, it adds the client Ids to the list and returns them. Condition: packageName is case-sensitive

The main method in the UserInterface class gets the total number of clients, and their details from the user.

Invoke the addClient method by passing the client details as colons separated and storing them in the clientSet which is a HashSet.

Get the packageName from the user and invoke the getTotalClientsByPackageName method, passing the packageName as a parameter to find the count of clients for the specified packageName. If no packageName is found, display: "No clients found with package name<packageName>", otherwise, print the count of clients as shown in the sample output.

Get the packageName from the user and and invloke the listClientIdsByPackageName` method, passing the packageName as a paramete to retrieve client Ids based on a specified pizzaType and add it to the list. If no packageName is found, display: "No clients found with package name <packageName>", otherwise, print the client Ids as shown in the sample output.

Note:

- 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 the 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/Output 1:

Enter number of clients to be added

3

Enter the client details

FIT0324:Silver

FIT5345:Gold

FIT5678:Silver

Enter the package name to find the total count of clients

Silver

Total number of clients with package name Silver is 2

Enter the package name to search for client Ids

Silver

Client Ids based on package name Silver

FIT0324

FIT5678

Sample Input/Output 2:

Enter number of clients to be added

3

Enter the client details

FIT4556:Silver

FIT3456:Gold

FIT2312:Bronze

Enter the package name to find the total count of clients

Platinum

No clients found with package name Platinum

Enter the package name to search for client Ids

Bronze

Client Ids based on package name Bronze

FIT2312

Sample Input/Output 3:

Enter number of clients to be added

2

Enter the client details

FIT4523:Gold

FIT3423:Silver

Enter the package name to find the total count of clients

Silver

Total number of clients with package name Silver is 1

Enter the package name to search for client Ids

Bronze

No clients found with package name Bronze

10.Visa Ledger

Sky Travels, a reliable travel agency in Ahmedabad, seeks to enhance its visa management system with the Visa Ledger. As the developer, your task is to implement the VisaInfo class with methods to add, count, and list visa details, ensuring case-insensitive handling of country names. This will streamline visa management, improving organisation and service quality.

Component Specification: VisaInfo Class

Type (Class) Attributes Methods

VisaInfo private Set<String> visaSet Getter and setter methods for the attribute are included in the code skeleton.

Note: The visaSet contains the visaNumber and countryName, separated by a colon (.

Requirements Type (Class) Methods Responsibilities

Inserts the visaDetails to visaSet. VisaInfo public void addVisaDetailsToSet(String visaDetails) This method takes visaDetails, a string with visaNumber and countryName separated by a colon (, as a parameter and inserts these details into the visaSet.

Find the count of visas based on the given countryName. VisaInfo public int countVisaNumbersForCountry(String countryName) This method accepts countryName as an argument. If the countryName matches the countryName present in the Map,

it must count the number of visaNumbers on the given countryName and return the same. Else return -1.

Condition: countryName is case-sensitive

Filter the visaNumber based on the countryName. VisaInfo public List<String> listVisaNumbersByCountry(String countryName) This method filters the countryName and returns the list of visaNumbers that have the same countryName.

Condition: countryName is case-sensitive

The main method in the UserInterface class gets the total number of visa, and their details from the user.Invoke the addVisaDetailsToSet method to add the details in to the visaSet.

Get the the countryName from the user and invoke countVisaNumbersForCountry method which counts and retrieves visa's associated with that country. Display the results by refer to sample input and output.

Get the the countryName from the user and invoke the listVisaNumbersByCountry method which retrieves visaNumbers associated with that country. Display the results by refer to sample input and output.

Note:

- 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 the 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/Output 1:

Enter the number of visa records to be added:

3

Enter the visa records:

A123:USA

B456:Canada

C789:USA

Enter the country name to be searched for visa count

USA

The visa count for USA is 2

Enter the country name to find visa numbers

Canada

Visa numbers based on Canada are

B456

Sample Input / Output 2:

Enter the number of visa records to be added:

2

Enter the visa records:

D101:UK

E202:Australia

Enter the country name to be searched for visa count

China

No visa numbers were found for China

Enter the country name to find visa numbers

Australia

Visa numbers based on Australia are

E202

Sample Input/Output 3:

Enter the number of visa records to be added:

3

Enter the visa records:

F435:Japan

G675:China

H230:Japan

Enter the country name to be searched for visa count

Japan

The visa count for Japan is 2

Enter the country name to find visa numbers

USA

No visa numbers were found for USA

12.Sport Event Scheduler

Sports Scheduler wants to create an advanced system to help users easily retrieve match details for specific coordinators. Additionally, the system ensures matches are listed in ascending order by match date and displays distinct match details for streamlined management. As a software developer, help them automate this process.

Requirements

1. Retrieve match details for the specified coordinator.
2. Retrieve match details in ascending order by matchDate.
3. Retrieve distinct matchTitle.

Component Specification: Match (POJO class)

Type(Class)	Attributes	Methods
Match	String matchCode	
	String matchTitle	
	String coordinator	
	LocalDate matchDate	
	String venue	Getters and setters, no argument, and five-argument constructors are given in the code skeleton.
	The code skeleton also includes toString method.	

Component Specification: MatchUtility

Type (Class)	Methods	Responsibilities
MatchUtility	public Stream<Match> getMatchesByCoordinator(List<Match> matchList, String coordinator)	This method takes a list of Match objects and a specific coordinator as input parameters. It filters the Matches in the list based on their coordinator name and returns a stream containing only the Matches for the specified coordinator.
	Condition: The coordinator is case-sensitive	
MatchUtility	public Stream<Match> getMatchesInOrderByDate(List<Match> matchList)	This method takes a list of Match objects as input, sorts them in ascending order based on their matchDate, and returns a stream of the sorted Matches.
MatchUtility	public Stream<String> getDistinctMatches(List<Match> matchList)	This method takes a list of Match objects as input and returns a stream containing only the unique matchTitle, removing any duplicates.

The main method in the UserInterface class gets the total number of Matches, and their details from the user are provided as a part of the code skeleton. Create an object for the Match class, set the values to the object, and store all the objects in a list.

- Get the coordinator from the user. Invoke the getMatchesByCoordinator method to filter the match by coordinator. If the matches are available for the given coordinator, then print the available match details using toString() method. Otherwise, print "No Matches found for the given Coordinator <coordinator>".

- Invoke the getMatchesInOrderByDate method to sort the matches in ascending order by matchDate, then print the result using toString() method as shown in sample input / output.
- Invoke the getDistinctMatches method to get the unique matchTitle, then print the result using toString() method as shown in sample input / output.

Note:

- 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.
- Assume that the number of matches needed to be entered into the list is always a valid positive number.
- Assume that the number of matches to be retrieved from the list is always a valid positive number.
- Assume that the matches details are always valid.
- The date should be displayed in the format yyyy-MM-dd.
- Do not use System.exit(x) to terminate the code.

Sample Input / Output 1

Enter the number of Matches:

4

Enter the Match details:

M4001:Rugby Match:Paul:12-03-2024:Twickenham Stadium

M4002:Volleyball Match:Anna:05-02-2024:Beach Arena

M4003:Golf Tournament:Tom:28-08-2024:Augusta National

M4001:Rugby Match:Paul:13-03-2024:Twickenham Stadium

Enter the Coordinator:

Paul

Matches for the given coordinator:

M4001 | Rugby Match | Paul | 2024-03-12 | Twickenham Stadium

M4001 | Rugby Match | Paul | 2024-03-13 | Twickenham Stadium

Matches in ascending order based on match date:

M4002 | Volleyball Match | Anna | 2024-02-05 | Beach Arena

M4001 | Rugby Match | Paul | 2024-03-12 | Twickenham Stadium

M4001 | Rugby Match | Paul | 2024-03-13 | Twickenham Stadium

M4003 | Golf Tournament | Tom | 2024-08-28 | Augusta National

Distinct Matches are:

Rugby Match

Volleyball Match

Golf Tournament

Sample Input/Output 2

Enter the number of Matches:

3

Enter the Match details:

M5001:Ice Hockey Match:Nick:18-11-2023:Bell Centre

M5002:Formula 1 Race:Lewis:29-09-2023:Silverstone Circuit

M5003:Horse Racing:Emily:15-10-2023:Churchill Downs

Enter the Coordinator:

Ronnie

No Matches found for the given Coordinator Ronnie

Matches in ascending order based on match date:

M5002 | Formula 1 Race | Lewis | 2023-09-29 | Silverstone Circuit

M5003 | Horse Racing | Emily | 2023-10-15 | Churchill Downs

M5001 | Ice Hockey Match | Nick | 2023-11-18 | Bell Centre

Distinct Matches are:

Ice Hockey Match

Formula 1 Race

Horse Racing

13.Prime Corp Employee Management

Prime Corp needs to develop an advanced system for HR managers to effortlessly access and manage employee details. The system should retrieve employee details based on the specified department, ensure employees are listed in ascending order by their joining date and include only unique employee roles for better management. As a software developer, help them automate this process.

Requirements

1. Retrieve employee details for the specified department.
2. Retrieve employee details in ascending order by joining date.
3. Retrieve unique employee roles.

Component Specification: Employee (POJO class)

Type(Class)	Attributes	Methods
Employee	String employeeId String employeeName String departmentName LocalDate joiningDate String role	Getters and setters, no argument, five-argument constructors are given in the code skeleton.

Component Specification: EmployeeUtility (Utility class)

Type (Class)	Methods	Responsibilities
EmployeeUtility	public Stream<Employee> getEmployeesByDepartment(List<Employee> employeeList, String departmentName)	This method takes a list of Employee details and a departmentName as parameters. It filters the employees in the list based on the departmentName and returns a stream containing only the employees from the specified departmentName. Condition: departmentName is case-sensitive.
EmployeeUtility	public Stream<Employee> getEmployeesInAscendingOrderByDate(List<Employee> employeeList)	This method takes a list of Employee details as input, sorts them in ascending order based on the joiningDate, and returns a stream of the sorted employee details.
EmployeeUtility	public Stream<String> getUniqueEmployeeRoles(List<Employee> employeeList)	

This method takes a list of Employee details as input and returns a stream containing only the unique employee roles, removing any duplicates.

The main method in the UserInterface class gets the total number of employees and their details from the user, provided as part of the code skeleton. Create an object for the Employee class, set the values to the object, and store all the objects in a list.

- Get the department name from the user and invoke the getEmployeesByDepartment method to filter the employees by department. If the employees are available for the given department, then print the available employee details as shown in the sample input/output. Otherwise, print "No Employees found for the given Department <department>".
- Invoke the getEmployeesInAscendingOrderByDate method to sort the employees in ascending order by joiningDate, then print the result as shown in the sample input/output.
- Invoke the getUniqueEmployeeRoles method to get the unique employee roles, then print the result as shown in the sample input/output.

Note:

- 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.
- Assume that the employee details are always valid.
- The date should be displayed in the format dd-MM-yyyy.
- Do not use System.exit(x) to terminate the code.

Sample Input/Output 1:

Enter the number of Employees

5

Enter the Employee details:

E001:John Doe:Finance:15-03-2020:Manager

E002:Jane Smith:IT:22-06-2018:Developer

E003:Mike Johnson:HR:12-11-2019:Recruiter

E004:John Doe:Finance:16-03-2020:Manager

E005:Emily Davis:IT:10-01-2021:Analyst

Enter the Department name:

IT

Employees for the given department:

Employee Name: Jane Smith, Joining Date: 22-06-2018

Employee Name: Emily Davis, Joining Date: 10-01-2021

Employees in ascending order based on joining date:

Employee Name: Jane Smith, Joining Date: 22-06-2018, Role: Developer

Employee Name: Mike Johnson, Joining Date: 12-11-2019, Role: Recruiter

Employee Name: John Doe, Joining Date: 15-03-2020, Role: Manager

Employee Name: John Doe, Joining Date: 16-03-2020, Role: Manager

Employee Name: Emily Davis, Joining Date: 10-01-2021, Role: Analyst

Unique Employee roles are:

Manager

Developer

Recruiter

Analyst

Sample Input/Output 2:

Enter the number of Employees

3

Enter the Employee details:

E005:Chris Brown:Marketing:12-08-2021:Coordinator

E006:Jessica Taylor:Finance:05-02-2019:Analyst

E007:Chris Brown:Marketing:11-08-2021:Coordinator

Enter the Department name:

Sales

No Employees found for the given Department Sales

Employees in ascending order based on joining date:

Employee Name: Jessica Taylor, Joining Date: 05-02-2019, Role: Analyst

Employee Name: Chris Brown, Joining Date: 11-08-2021, Role: Coordinator

Employee Name: Chris Brown, Joining Date: 12-08-2021, Role: Coordinator

Unique Employee roles are:

Coordinator

Analyst

13.Footwear Heaven

Footwear Haven aims to create an advanced system to help store managers easily access and manage shoe inventory details. The system should manipulate and retrieve shoe details for a specified brand, ensuring the shoes are listed in ascending order by size and include only unique shoe names for more efficient management. As a software developer, your task is to automate this process.

Requirements

1. Retrieve shoe details for the specified brandName.
2. Retrieve shoe details in ascending order by size.
3. Retrieve unique shoeNames.

Component Specification: ShoeDetails (POJO class)

Type(Class)	Attributes	Methods
ShoeDetails	String shoeId	
	String shoeName	
	String brandName	
	LocalDate arrivalDate	
	int size Getters and setters, no argument, and five-argument constructors are given in the code skeleton.	

The code skeleton also includes toString method.

Component Specification: ShoeUtility (Utility class)

Type (Class)	Methods	Responsibilities
ShoeUtility	public Stream<ShoeDetails> fetchShoesByBrand(List<ShoeDetails> shoeList, String brandName)	This method takes a list of ShoeDetails and a brandName as input parameters. It filters the ShoeDetails in the list based on the brandName and returns a stream containing only the ShoeDetails of the specified brandName. Condition: brandName is case-insensitive.
ShoeUtility	public Stream<ShoeDetails> fetchShoesInAscendingOrderBySize(List<ShoeDetails> shoeList)	This method takes a list of ShoeDetails as input parameter and sort them in ascending order based on the size, and returns a stream of the sorted shoe details.
ShoeUtility	public Stream<String> fetchUniqueShoes(List<ShoeDetails> shoeList)	This method takes a list of ShoeDetails as an input parameter and returns a stream that contains only the unique shoeNames, removing any duplicates.

The main method in the UserInterface class gets the total number of shoes and their details from the user, provided as part of the code skeleton. Create an object for the ShoeDetails class, set the values to the object, and store all the objects in a list.

- Get the brandName from the user.
- Invoke the fetchShoesByBrand method to filter the ShoeDetails by brandName. If the shoes are available for the given brandName, then print the available shoe details as shown in the sample input/output. Otherwise, print "No Shoes found for the given Brand <brand>".
- Invoke the fetchShoesInAscendingOrderBySize method to sort the ShoesDetails in ascending order by size, then print the result as shown in the sample input/output.
- Invoke the fetchUniqueShoes method to get the unique shoeNames, then print the result as shown in the sample input/output.

Note:

- 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.
- Assume that the shoe details are always valid.
- Do not use System.exit(x) to terminate the code.

Sample Input/Output 1:

Enter the number of Shoe details:

4

Enter the Shoe details:

S001:Nike Air Max:Nike:15-03-2021:10

S002:Adidas Ultra Boost:Adidas:22-06-2019:9

S003:Puma Suede:Puma:12-11-2020:8

S004:Classic:Nike:10-01-2022:11

Enter the Brand name:

Nike

Shoes for the given brand:

ShoeDetails [shoeId:S001, shoeName:Nike Air Max, brandName:Nike, arrivalDate:2021-03-15, size:10]

ShoeDetails [shoeId:S004, shoeName:Classic, brandName:Nike, arrivalDate:2022-01-10, size:11]

Shoes in ascending order based on size:

ShoeDetails [shoeId:S003, shoeName:Puma Suede, brandName:Puma, arrivalDate:2020-11-12, size:8]

ShoeDetails [shoeId:S002, shoeName:Adidas Ultra Boost, brandName:Adidas, arrivalDate:2019-06-22, size:9]

ShoeDetails [shoeId:S001, shoeName:Nike Air Max, brandName:Nike, arrivalDate:2021-03-15, size:10]

ShoeDetails [shoeId:S004, shoeName:Classic, brandName:Nike, arrivalDate:2022-01-10, size:11]

Unique Shoe names are:

Nike Air Max

Adidas Ultra Boost

Puma Suede

Classic

Sample Input/Output 2:

Enter the number of Shoe details:

3

Enter the Shoe details:

S005:Nike Pegasus:Nike:11-11-2017:9

S006:New Balance 574:New Balance:19-05-2018:10

S007:Nike Pegasus:Nike:11-11-2017:8

Enter the Brand name:

Puma

No Shoes found for the given Brand Puma

Shoes in ascending order based on size:

ShoeDetails [shoeId:S007, shoeName:Nike Pegasus, brandName:Nike, arrivalDate:2017-11-11, size:8]

ShoeDetails [shoeId:S005, shoeName:Nike Pegasus, brandName:Nike, arrivalDate:2017-11-11, size:9]

ShoeDetails [shoeId:S006, shoeName:New Balance 574, brandName:New Balance, arrivalDate:2018-05-19, size:10]

Unique Shoe names are:

Nike Pegasus

New Balance 574

14Workshop Wizard

You are part of a software development team working on an Employee Training Management System. As your organization grows, keeping track of training programs, trainers, and their details becomes increasingly challenging. To streamline training management and improve team productivity, your organization decides to develop an Employee Training Management System. Your role is to assist in automating training management processes based on predefined criteria.

Develop a Java application using streams to meet the requirements efficiently.

Requirements:

1. Retrieve the training programs based on trainer name.
2. Retrieve the training programs based on topic.
3. Retrieve the training programs based on duration.

Component Specification: TrainingProgram (POJO Class)

Type (Class) Attributes Methods

TrainingProgram String programName

String trainerName

String topic

int duration Getters and setters, no argument, and four-argument constructors are given in the code skeleton.

Component Specification: TrainingProgramUtility

Type (Class) Methods Responsibilities

TrainingProgramUtility public List<TrainingProgram> retrieveProgramsByTrainer(Stream<TrainingProgram> programStream, String trainerName) This method takes a stream of TrainingProgram objects and a trainerName as parameters. It filters the stream to include only those programs whose trainer name matches the specified trainer name and collects the filtered programs into a list and returns it.

Condition: trainerName is case-sensitive.

TrainingProgramUtility public List<TrainingProgram> retrieveProgramsByTopic(Stream<TrainingProgram> programStream, String topic) This method takes a stream of TrainingProgram objects and a topic as parameters.It filters the stream to include only those programs whose topic matches the specified topic and collects the filtered programs into a list and returns it.

TrainingProgramUtility public List<TrainingProgram> retrieveProgramsByDuration(Stream<TrainingProgram> programStream, int duration) This

method takes a stream of TrainingProgram objects and a duration as parameters.It filters the stream to include only those programs whose duration is higher than the specified duration limit and collects the filtered programs into a list and returns it.

The main method in the UserInterface class interacts with the user to manage TrainingProgram in the System. It prompts the user to input the total number of programs and their details, creates TrainingProgram objects, sets their values, and stores them in a list.

Retrieve Programs by Trainer: The user is prompted to provide the name of the trainer. The retrieveProgramsByTrainer() method is called to filter programs by the specified trainerName. If programs are found, their details are printed in the format: Program Name:<programName>/Topic:<topic>/Duration:<duration>. If no programs are found, a message indicating "No programs found for the given trainer" should be displayed.

Retrieve Programs by Topic: The user is prompted to provide the topic. The retrieveProgramsByTopic() method is invoked to filter programs by the specified topic. If programs are found, their details are printed in the format: Program Name:<programName>/Trainer Name:<trainerName>/Duration:<duration>. If no programs are found, a message stating "No programs found with the given topic " should be displayed.

Retrieve Programs by Duration: The user is prompted to provide the duration. The retrieveProgramsByDuration() method is invoked to filter programs exceeding the specified duration. If programs are found, their details are printed in the format: Program Name:<programName>/Trainer Name:<trainerName>/Topic:<topic>/Duration:<duration>. If no programs are found, a message stating "No programs found with duration greater than the given number" should be displayed.

Note:

- 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 that you follow the object-oriented specifications provided in the question description.
- Ensure to provide the names for the classes, attributes, and methods as specified in the question description.
- Adhere to the code template, if provided.
- Please don't use System.exit(0) to terminate the program.

Sample Input / Output 1

Enter the total number of training programs to add to the list

5

Enter the training program details (programName, trainerName, topic, duration)

JavaScript Essentials,Toby Flenderson,Programming,10

Effective Negotiation,Phyllis Vance,Management,14

Cloud Architecture,Darryl Philbin,IT,18

Data Analysis,Creed Bratton,Data Science,20

Digital Marketing,Meredith Palmer,Marketing,16

Enter the trainer name

Darryl Philbin

Training programs conducted by Darryl Philbin:

Program Name:Cloud Architecture/Topic:IT/Duration:18

Enter the topic

IT

Training programs with topic 'IT':

Program Name:Cloud Architecture/Trainer Name:Darryl Philbin/Duration:18

Enter the minimum duration (in hours)

15

Training programs with duration greater than 15 hours:

Program Name:Cloud Architecture/Trainer Name:Darryl Philbin/Topic:IT/Duration:18

Program Name:Data Analysis/Trainer Name:Creed Bratton/Topic:Data Science/Duration:20

Program Name:Digital Marketing/Trainer Name:Meredith Palmer/Topic:Marketing/Duration:16

Sample Input / Output 2

Enter the total number of training programs to add to the list

4

Enter the training program details (programName, trainerName, topic, duration)

Conflict Management,Stanley Hudson,Management,9

Advanced Python,Oscar Martinez,Programming,15

Creative Thinking,Pam Beesly,Personal Development,7

Team Collaboration,Ryan Howard,Management,14

Enter the trainer name

Dwight Schrute

No programs found for the given trainer

Enter the topic

IT

No programs found with the given topic

Enter the minimum duration (in hours)

16

No programs found with duration greater than the given number