Highest Placement

SRV college wants to recognize the department which has succeeded in getting the maximum number of placements for this academic year. The departments that have participated in the recruitment drive are CSE,ECE, MECH. Help the college find the department getting maximum placements. Check for all the possible output given in the sample snapshot

Note : If any input is negative, the output should be "Input is Invalid". If all department has equal number of placements, the output should be "None of the department has got the highest placement".

Sample Input 1:

Enter the no of students placed in CSE:90
Enter the no of students placed in ECE:45
Enter the no of students placed in MECH:70
Sample Output 1:
Highest placement
CSE

Sample Input 2:
Enter the no of students placed in CSE:55
Enter the no of students placed in ECE:85
Enter the no of students placed in MECH:85
Sample Output 2:
Highest placement
ECE
MECH

Sample Input 3:
Enter the no of students placed in CSE:0
Enter the no of students placed in ECE:0
Enter the no of students placed in MECH:0
Sample Output 3:
None of the department has got the highest placement

Sample Input 4:
Enter the no of students placed in CSE:10
Enter the no of students placed in ECE:-50
Enter the no of students placed in MECH:40

Sample Output 3:
Input is Invalid

Code:

```
import java.util.*;

public class Placement{

  public static void main(String[] arg){

    Scanner Input = new Scanner(System.in);

    System.out.println("Enter the no of students placed in CSE:");

    int CSE = Input.nextInt();

    System.out.println("Enter the no of students placed in ECE:");

    int ECE = Input.nextInt();

    System.out.println("Enter the no of students placed in MECH:");

    int MECH = Input.nextInt();


    if(CSE==ECE && ECE==MECH && CSE==MECH){

      System.out.println("None of the department has got the highest placement");

    }

    else if(CSE>=0 && ECE>=0 && MECH>=0){

      if(MECH>ECE && MECH>CSE){

        System.out.println("Highest placement \nMECH");

      }

      else if(CSE>MECH && CSE>ECE){

      System.out.println("Highest placement \nCSE");

      }

      else if(ECE>MECH && ECE>CSE){

        System.out.println("Highest placement \nECE");

      }

      else if(ECE==MECH){
        System.out.println("Highest Placement \nECE \nMECH");
      }
      else if(CSE==ECE){
        System.out.println("Highest Placement \nCSE \nECE");
      }
      else if(CSE==MECH){
        System.out.println("Highest Placement \nCSE \nMECH");
      }

    }
    else{
      System.out.println("Input is invalid");
    }

  }
}
```

Fuel Consumption Calculator

Write a program to calculate the fuel consumption of your vehicle.

The program should ask the user to enter the quantity of petrol to fill up the tank and the distance covered till the tank goes dry.

Calculate the fuel consumption and display it in the format (liters per 100 kilometers).

Convert the same result to the U.S. style of miles per gallon and display the result. If the quantity or distance is zero or negative display "<respective_input> is an Invalid Input".

[Note: The US approach of fuel consumption calculation (distance / fuel) is the inverse of the European approach (fuel / distance). Also note that 1 kilometer is 0.6214 miles, and 1 liter is 0.2642 gallons.]

The result should be with two decimal place.

To get two decimal place refer the below-mentioned print statement :

float cost=670.23;

System.out.printf("You need a sum of Rs.%.2f to cover the trip",cost);

Sample Input 1:

Enter the no of liters to fill the tank

20

Enter the distance covered

150

Sample Output 1:

Liters/100KM

13.33

Miles/gallons

17.64

Explanation:

For 150 KM fuel consumption is 20 liters,

Then for 100 KM fuel consumption would be (20/150)*100=13.33,

Distance is given in KM, we have to convert it to miles (150*0.6214)=93.21,

Fuel consumption is given in liters, we have to convert it to gallons (20*0.2642)=5.284,

Then find (miles/gallons)=(93.21/5.284)=17.64

Sample Input 2:

Enter the no of liters to fill the tank

-5

Sample Output 2:

-5 is an Invalid Input

Sample Input 3:

Enter the no of liters to fill the tank

25

Enter the distance covered

-21

Sample Output 3:

-21 is an Invalid Input

Display Characters

Rohan wants a magic board, which displays a character for a corresponding number for his science exhibition. Help him to develop such application.

For example when the digits 65,66,67,68 are entered, the alphabet ABCD are to be displayed.

[Assume the number of inputs should be always 4]

Sample Input 1:
Enter the digits:
65
66
67
68

Sample Output 1:
65-A
66-B
67-C
68-D

Sample Input 2:
Enter the digits:
115
116
101
112

Sample Output 2:

```
115-s
116-t
101-e
112-p
```

String Concatenation

The authority of XYZ gated residential colony wants its residents' name datum Should be stored in the following format - residents' name <space> his/her father's name. Write a program to concat the father's name to the residents' name. The name should be validated,on validation the name should contain only alphabets and space is allowed. If the name is not valid display the message "Invalid name". If valid string then convert it to uppercase and print it..

[Use concat(String s) of the String class.]

Sample Input 1:

Inmate's name:Aron

Inmate's father's name:Terby

Sample Output 1:

```
ARON TERBY
```

Sample Input 2:

Inmate's name:Mary Anto

Inmate's father's name:Jose

Sample Output 2:

MARY ANTO JOSE

Sample Input 3:

Inmate's name:Dev12

Inmate's father's name:Terby

Sample Output 3:

Invalid name

Least offer

Maya buys "N" no of products from a shop. The shop offers a different percentage of discount on each item. She wants to know the item that has the minimum discount offer, so that she can avoid buying that and save money.

[Input Format: The first input refers to the no of items; the second input is the item name, price and discount percentage separated by comma(,)]

Assume the minimum discount offer is in the form of Integer.

Note: There can be more than one product with a minimum discount.

Sample Input 1:

4

mobile,10000,20

shoe,5000,10

watch,6000,15

laptop,35000,5

Sample Output 1:

shoe

Explanation: the discount on the mobile is 2000, the discount on the shoe is 500, the discount on the watch is 900 and the discount on the laptop is 1750. So the discount on the shoe is the minimum.

Sample Input 2:

```
4

Mobile,5000,10

shoe,5000,10

WATCH,5000,10

Laptop,5000,10
```

Sample Output 2:

```
Mobile

shoe

WATCH

Laptop
```

Ticket Price Calculation - Static

Ticket Calculation

Create a class Ticket with the following private variables
int ticketid;
int price;
static int availableTickets;

Include getters and setters methods in the Ticket class.

AvailableTickets should hold only positive value. Zero and negative values are not allowed.(This logic should be checked inside the corresponding setter method)

Write the following method in the Ticket class:

public int calculateTicketCost(int nooftickets) â€”this method should check the ticket availability, If the tickets are available, reduce the nooftickets from availableTickets and calculate the total amount as nooftickets*price and return the total amount. If the tickets are not available, this method should return -1.

Write a main method in the Main class to test the application.

Sample input and output

```
Enter no of bookings:
2
Enter the available tickets:
25
Enter the ticketid:
123
Enter the price:
100
Enter the no of tickets:
5
Available tickets: 25

Total amount:500

Available ticket after booking:20

Enter the ticketid:
124
Enter the price:
100
Enter the no of tickets:
2
Available tickets: 20

Total amount:200

Available ticket after booking:18
```

Student Details - Constructor

Create a class Student with the private attributes

int studentId

String studentName, studentAddress, collegeName.

Include appropriate getter methods.

Write 2 constructors for the Student class based on the below assumptions.

Assume most of the students are from "NIT" college. So user has to give input whether the student is from NIT or not.

1. If student belongs to NIT, give input as 'yes/YES' and skip input for the attribute collegeName and create student object with 3-argument constructor to initilze the values for studentId, studentName and studentAddress and collegeName as "NIT".
2. If student belongs to other college, give input as 'no/NO' and get college name from the user and create student object with 4-argument constructor to initialize all the values.
3. Instead of Yes / No, if user enters different input then display 'Wrong Input' and get the input again.

Based on the above assumptions write the necessary constructors in the Student class.

Write a class StudentMain with the main method and test the application.

Get all the input needed from the main method.

Sample Input 1:

Enter Student's Id:

12

Enter Student's Name:

John

Enter Student's address:

Chennai

Whether the student is from NIT(Yes/No):

NO

Enter the college name:

SVS

Sample Output 1:

```
Student id:12

Student name:John

Address:Chennai

College name:SVS
```

Sample Input 2:

```
Enter Student's Id:

43

Enter Student's Name:

Tom

Enter Student's address:

Coimbatore

Whether the student is from NIT(Yes/No):

y

Wrong Input

Whether the student is from NIT(Yes/No):

yes
```

Sample Output 2:

```
Student id:43

Student name:Tom

Address:Coimbatore
```

College name:NIT

BankAccountDetails

In the first round of HR interview for a banking sector, HR decides to make candidates design an application which provides only information on transaction like amount withdrawn with respect to fields given. Develop a program to implement this scenario.

Create a class Account with the private attributes:

- accountId int
- accountType String
- balance int

The method **public boolean withdraw(int)** used to calculate the current balance of the respective account. Before that it should enough balance. If there is enough balance, deduct the amount from the balance and print "Balance amount after withdraw: XXX" and return true. If there is no enough balance, print "Sorry!!! No enough balance" and return false.

Create a class AccountDetails with main function and the below methods :

- public Account getAccountDetails() - This methods gets the input related to Account from the user and returns the Account object with all values set. If the input given for balance is less than or equal to zero, consider it as invalid and display "Balance should be positive". Continue this kind of evaluation till user enters a positive value.

- public int getWithdrawAmount() - This methods gets the amount to be withdrawn as input from the user and returns the same. If the input given for amount is less than or equal to zero, consider it as invalid and display "Amount should be positive". Continue this kind of evaluation till user enters a positive value.

Use appropriate getters and setters.

Note: For successful compilation and evaluation of the code , use a static Scanner inside the AccountDetails class.

Sample input 1:

Enter account id:

100
Enter account type:

Savings

Enter balance:

10000
Enter amount to be withdrawn:

500

Sample Output 1:

Balance amount after withdraw: 9500

Sample input 2:
Enter account id:

101
Enter account type:

Savings
Enter balance:
1000
Enter amount to be withdrawn:
1500

Sample Output 2:

Sorry!!! No enough balance

Sample input 3:

Enter account id:

100

Enter account type:

Savings

Enter balance:

-100

Balance should be positive

Enter balance:

5000

Enter amount to be withdrawn:
500

Sample Output 1:

Balance amount after withdraw: 4500

Contact Details of Hosteller

SNMR College of Engineering and Technology wants to create an application to store their students details as well as the details of hostellers.

In case of any changes to be made to the attributes, admin can update the details like room number and phone number of the hosteler.

Develop a program to implement this scenario.

Create a public class Student with protected attributes :

int studentId

String name

int departmentId

String gender

String phone

Create a public class Hosteller with private attributes

String hostelName

int roomNumber

Make this class inherit the Student class, as it holds all the properties of Student.

Use appropriate public getters and setters for both the classes.

Write a class Main with the main function.

In Main class get the input of the hosteller using the method :

public static Hosteller getHostellerDetails().

Invoke this method from the main method and then modify the room number and phone number, if needed.

Sample Input 1:

```
Enter the Details:
Student Id
1
Student Name
John
Department Id
101
Gender
Male
Phone Number
9876543210
Hostel Name
YMCA
Room Number
10
Modify Room Number(Y/N)
Y
New Room Number
11
Modify Phone Number(Y/N)
Y
New Phone Number
9876543121
```

Sample Output 1:

```
The Student Details
1 John 101 Male 9876543121 YMCA 11
```

Sample Input 2:

```
Enter the Details:
Student Id
2
Student Name
John Paul
Department Id
112
Gender
Male
Phone Number
9885526536
Hostel Name
YMBA
Room Number
5
Modify Room Number(Y/N)
N
Modify Phone Number(Y/N)
N
```

Sample Output 2:

```
The Student Details:
2 John Paul 112 Male 9885526536 YMBA 5
```

Employee Loan Eligibility - Polymorphism

Global Engineering is one of the fastest growing company. It needs to automate the transactions performed in the organization. As start up, they need to automate the Employee management system.

Partial code is provided to do this. Don't change the skeleton. Do the additions wherever necessary.

You are provided with a public class Employee with protected attributes :

int employeeId

String employeeName

double salary

Appropriate public getters and setters are already written.

Write a public 2 argument constructor with arguments - employeeId,and employeeName.

Write a public abstract method calculateSalary() in Employee class as,

public abstract void calculateSalary()

You are provided with a public class PermanentEmployee with private attribute :

double basicPay

Appropriate public getters and setters are already written.

Make this class PermanentEmployee to inherit the Employee class.

Write a public 3 argument constructor with arguments - employeeId, employeeName and basicPay.

Implement the calculateSalary method in Employee class as

salary = basicPay - PF amount

Here PF Amount = basicPay * 0.12; Set this value to the salary attribute.

You are provided with a public class TemporaryEmployee with private attribute :

int hoursWorked

int hourlyWages

Appropriate public getters and setters are already written.

This class TemporaryEmployee should inherit the Employee class.

Write a public 4 argument constructor with arguments - employeeId, employeeName, hoursWorked and hourlyWages.

Implement the calculateSalary method in Employee class as

salary = hoursWorked * hourlyWages

Set this value to the salary attribute.

You are provided with a public class Loan

A method calculateLoanAmount is provided as shown below :

public double calculateLoanAmount(Employee employeeObj)

This method should calculate the loan amount and return that amount.

Provide the implementation for this method as mentioned below

Loan amount is calculated as follows :

If the Employee object is of type PermanentEmployee the loan amount should be 15% of the salary.

If the Employee object is of type TemporaryEmployee the loan amount should be 10% of the salary.

You are provided with a public class Main which has the main method.

Check the correctness of the methods written in these classes.

Note : All class, methods needs to be declared as public

Vehicle-Loan-Insurance - Use Interface

Universal Loan and Insurance Providers is one of the fastest growing organization. It needs to automate the transactions performed in the organization. They need to automate the process of issuing loan and insurance coverage for vehicles.

Partial code is provided to do this. Don't change the skeleton. Do the additions wherever necessary.

You are provided with a public class Vehicle with private attributes :

String vehicleNumber

String modelName

String vehicleType

double price

Appropriate public getters and setters are already written.

You are also provided with a 4 argument constructor with arguments -vehicleNumber, modelName, vehicleType and price.

Note that the vehicleType can take the values as "4 wheeler" or "3 wheeler" or "2 wheeler".

Write a public interface Loan with an abstract method "double issueLoan()".

Write a public interface Insurance with an abstract method "double takeInsurance ()".

The above class Vehicle should implement the Interfaces Loan and Insurance.

Provide the implementation for issueLoan method based on the type of Vehicle.

If the vehicleType is "4 wheeler", the eligible loan amount is 80% of its price.

If the vehicleType is "3 wheeler", the eligible loan amount is 75% of its price.

If the vehicleType is "2 wheeler", the eligible loan amount is 50% of its price.

Provide the implementation for takeInsurance based on price of vehicle.

If the vehicle price is less than or equal to 150000 insurance amount is 3500.

If the vehicle price is greater than 150000 and less than or equal to 300000 insurance amount is 4000.

If the vehicle price is greater than 300000 insurance amount is 5000.

You are provided with a public class Main which has the main method.

Check the correctness of the methods written in these classes.

Account Manipulation - Abstract class

Yzee bank needs to automate the bank transactions. There are many accounts, like Savings Account, Current Account, Demat Account and so on.

As start up, they need to automate the Savings Account details.

You are provided with a public class Customer with private attributes :

int customerId

String customerName

String emailId

Appropriate public getters and setters are already written.

Write a public 3 argument constructor with arguments - customerId, customerName and emailId.

Write a public class Account with protected attributes :

int accountNumber

Customer customerObj

double balance

Uncomment the public getters and setters provided in the template.

Write a public 3 argument constructor with arguments - accountNumber, customerObj and balance.

Write a public method in Account class as,

public boolean withdraw(double amount) - Make this method as abstract.

Write a public class SavingsAccount with private attribute :

double minimumBalance

Uncomment the public getters and setters provided in the template.

Make this class SavingsAccount to inherit the Account class.

Write a public 4 argument constructor with arguments - accountNumber, customerObj, balance and minimumBalance.

Implement the withdraw method as

public boolean withdraw(double amount) - This method should return true if withdraw is successful, else return false.

In this method, check if

balance - amount is greater than the minimum balance.

If yes, perform withdraw. Reduce the withdraw amount from the balance and return true.

If not, return false.

Create a public class Main which has the main method. Check the correctness of the methods written in these classes.

Note : All class, methods needs to be declared as public

Register a Candidate - User defined Exception(with throw and throws)

Geneva Technologies is planning to conduct a Walk-in interview. The interview has 4 levels. To attend the interview, the candidates need to register the following information: Name, Gender and Expected salary.

Help him do this by writing a java program.

Partial code is provided.

You are provided with a public class Candidate with private attributes :

String name

String gender

double expectedSalary

Appropriate getter and setters are provided.

You are provided with a public class Main.

Write a method getCandidateDetails as -

public static Candidate getCandidateDetails() - This method should get the candidate details, create the Candidate object using those details and return that object.

If the candidate's expected salary is less than 10000

- throw a user defined exception as InvalidSalaryException with the message "Registration Failed. Salary cannot be less than 10000." and return null.
- this method should throw / propagate InvalidSalaryException.

To do this, write a class InvalidSalaryException that inherits Exception class.

Write a constructor that takes a String as argument and set this string to the message attribute of the super class, Exception.

In the Main class, write the main method and test the method getCandidateDetails.

If it returns a valid Candidate object, then display "Registration Successful".

Use a catch block to handle the exception that is returned by the method getCandidateDetails. In catch block display the message by using the getMessage() method.

Sample Input 1:

Enter the candidate Details

Name

Margrett

Gender

Female

Expected Salary

50000

Sample Output 1:

Registration Successful

Sample Input 2:

Enter the candidate Details

Name

Robin

Gender

Male

Expected Salary

5000

Sample Output 2:

Registration Failed. Salary cannot be less than 10000.

Array Manipulation - Use try with multi catch

Tom wants to store the price details of the products that he purchased from the departmental store. Help him do this by using the concept of Arrays.

To do this create a public class ArrayException with a method getPriceDetails as :

public String getPriceDetails() - This method should do the following

Get the size of an array as input and then get the elements of the array(all elements are int) as input.

Next, user should provide the index of the array. This method should return the element at that index as "The array element is "+<that value>

This program may generate ArrayIndexOutOfBoundsException / InputMismatchException

In case of ArrayIndexOutOfBoundsException, the function should return "Array index is out of range".

When providing the input, if the input is not an integer, it will generate InputMismatchException. In this case the function should return "Input was not in the correct format".

Use exception handling mechanism to handle the exception. Use separate catch block for handling each exception. In the catch block, return the appropriate message.

Write a main method and test the above function.

Sample Input 1:

Enter the number of elements in the array
5
Enter the price details
50

80

60

70

40
Enter the index of the array element you want to access
1

Sample Output 1:

The array element is 80

Sample Input 2:

Enter the number of elements in the array
2
Enter the price details
50
80
Enter the index of the array element you want to access
9

Sample Output 2:

Array index is out of range

Sample Input 3:

Enter the number of elements in the array
2
Enter the price details
30

j

Sample Output 3:

Input was not in the correct format

Number of New Words

Miss.Jane, an experienced English professor, gives practice tests to her students to improve their written skills. Everyday students write an article and they submit it to Jane. Jane is particular that the students use only special characters like , ; : . ? ! in the article.

She counts the total number of words used and the number of new words used by the students in the article. Based on the analysis done on the new words used by the students she gives her feedback to the students.

Jane finds it difficult when the number of students increase. So she wanted to automate the process in the following format. Help her to write a java program to display the new words using lower case and in alphabetical order.

Sample Input and Output - 1

Enter Student's Article

Hello Everybody, welcome to collection in JAVA. Collection, is like a container and powerful concept in Java!

Number of words 17

Number of unique words 14

The words are

1. a

2. and

3. collection

4. concept

5. container

6. everybody

7. hello

8. in

9. is

10. java

11. like

12. powerful

13. to

14. welcome

Sample Input and Output - 2

Enter Student's Article

hello Hello HEllo hi hi: hi! Welcome,   welcome

Number of words 8

Number of unique words 3

The words are

1. hello

2. hi

3. welcome

Insurance Bazaar

Insurance Bazaar

Insurance Bazaar is developing an online website for showcasing various types of Insurance policies to their customers based on their needs. There are various types of Insurances provided by different insurance agencies. The admin of Insurance Bazaar wants to add different insurance policy names like Max Bupa Health Insurance, SBI

Health Insurance, IFFCO Tokio Two Wheeler Insurance and New India Assurance Two Wheeler Insurance to his database with Policy ID as the Tags.

Customers can view the names of all the polices available in Insurance Bazaar based on the type of insurance.

Write a Java program to simulate this scenario. Key (Policy ID) should be an Integer and Value (Insurance policy name) should be a String. The key-value should be sorted based on the key. Use the appropriate Collection to Store all these details and display.

The **addPolicyDetails** method should add the Policy ID and the Policy name into the appropriate map.

The **searchBasedOnPolicyType** method should return the list of Insurance policy ID depending on the input provided. This method takes the input as string (Input can be either Health or Two Wheeler).

The signature of the above functions are given as part of code skeleton, do not change the function signature.

Sample Policy ID and policy names:

Policy ID	**policy names**
10654	Max Bupa Health Insurance
10321	SBI Health Insurance
20145	IFFCO Tokio Two Wheeler Insurance
20165	New India Assurance Two Wheeler Insurance
10110	Reliance Health Insurance

Sample Input and Output1

Enter the no of Policy names you want to store

5

Enter the Policy ID

10654

```
Enter the Policy Name
Max Bupa Health Insurance
Enter the Policy ID
10321
Enter the Policy Name
SBI Health Insurance
Enter the Policy ID
20145
Enter the Policy Name
IFFCO Tokio Two Wheeler Insurance
Enter the Policy ID
20165
Enter the Policy Name
New India Assurance Two Wheeler Insurance
Enter the Policy ID
10110
Enter the Policy Name
Reliance Health Insurance
10110 Reliance Health Insurance
10321 SBI Health Insurance
10654 Max Bupa Health Insurance
20145 IFFCO Tokio Two Wheeler Insurance
20165 New India Assurance Two Wheeler Insurance
```

Enter the policy type to be searched

Two Wheeler

20145

20165

Sample Input and Output2

Enter the no of Policy names you want to store

4

Enter the Policy ID

10654

Enter the Policy Name

Max Bupa Health Insurance

Enter the Policy ID

10321

Enter the Policy Name

SBI Health Insurance

Enter the Policy ID

20145

Enter the Policy Name

IFFCO Tokio Two Wheeler Insurance

Enter the Policy ID

20165

Enter the Policy Name

New India Assurance Two Wheeler Insurance

10321 SBI Health Insurance

10654 Max Bupa Health Insurance

20145 IFFCO Tokio Two Wheeler Insurance

20165 New India Assurance Two Wheeler Insurance

Enter the policy type to be searched

Health

10321

10654

PhoneBook Manipulation

Airone mobile services needs to store their customer details in the company portal. The details are customer's first and last name, phone number, and email address. Help the company develop an application to maintain the details of their customer systematically.

You are provided with a class Contact with the following attributes as private.

String firstName

String lastName

long phoneNumber

String emailId

A 4 argument constructor and appropriate setters and getters to store and retrieve the details are also provided.

Create a class PhoneBook with a private attribute

List<Contact> phoneBook = new ArrayList<Contact>();

Write the getters and setters.

Write the following methods in the PhoneBook class.

public void addContact(Contact contactObj) - This method should add the contact object to the phoneBook list.

public List<Contact> viewAllContacts() - This method should return the list of all contacts available.

public Contact viewContactGivenPhone(long phoneNumber) - This method should return the contact details which has the phoneNumber given as parameter.

public boolean removeContact(long phoneNumber) - This method should remove the contact details which has the phoneNumber given as parameter. If removed return true. Else if the phone number is not available return false.

Write a class Main with the main method. Create the menu as shown in the Sample Input and Output and invoke the corresponding methods as per the choice provided.

Sample Input and Output 1:

Menu

1.Add Contact

2.Display all contacts

3.Search contact by phone

4.Remove contact

5.Exit

Enter your choice: 1

Add a Contact:

Enter the First Name: John

Enter the Last Name: Michael

```
Enter the Phone No.: 9787878900

Enter the Email: John@gmail.com

Menu

1.Add Contact

2.Display all contacts

3.Search contact by phone

4.Remove contact

5.Exit

Enter your choice: 1

Add a Contact:

First Name: Jhonty

Last Name: Rhodes

Phone No.: 9787888900

Email: Jhonty@gmail.com

Menu

1.Add Contact

2.Display all contacts

3.Search contact by phone

4.Remove contact

5.Exit

Enter your choice: 2

The contacts in the List are:

First Name: John
```

```
Last Name: Michael

Phone No.: 9787878900

Email: John@gmail.com

First Name: Jhonty

Last Name: Rhodes

Phone No.: 9787888900

Email: Jhonty@gmail.com

Menu

1.Add Contact

2.Display all contacts

3.Search contact by phone

4.Remove contact

5.Exit

Enter your choice: 3

Enter the Phone number to search contact:9787888900

The contact is:

First Name: Jhonty

Last Name: Rhodes

Phone No.: 9787888900

Email: Jhonty@gmail.com

Menu

1.Add Contact

2.Display all contacts
```

```
3.Search contact by phone

4.Remove contact

5.Exit

Enter your choice: 4

Enter the Phone number to remove :9787888900

Do you want to remove the contact (Y/N): Y

The contact is successfully deleted.

Menu

1.Add Contact

2.Display all contacts

3.Search contact by phone

4.Remove contact

5.Exit

Enter your choice: 5
```

Auditing

Lego is a famous company in the city. The company is due for auditing in the upcoming days. During the auditing details of the employees will be verified, but only for selected employees. Manually collecting the details will be difficult. So help them to print the details of the employees as per the requirement. The details of the employees need to be printed according to their salary.

Help them to write a java program to find the employees whose salary is less than or equal to the salary specified by the auditors using Lambdas.

Requirement 1: Find the employee details

Lego company wants to identify the employees whose salary is less than or equal to the salary specified by the auditors. By using the method fetchEmployeeDetails, identify the employees whose salary is less than or equal to the salary specified by the auditors.

Component Specification: Employee Interface- This is a Functional Interface.

Type(Interface)	**Methods**	**Responsibilities**
EmployeeAudit	public ArrayList<String> fetchEmployeeDetails (double salary)	This method is used to identify the employees whose salary is less than or equal to the salary passed as an argument by using Lambda expressions.

Component Specification: Main Class

- Add the employee details using a Map which holds the key as Employee name and value as Employees' salary. Where the map is given as a private attribute with the getters and setters as a part of the code skeleton.

- Then implement the method public static EmployeeAudit findEmployee() which holds the employee details using Lambda Expressions.

Component Name	**Type(Class)**	**Methods**	**Responsibilities**
Obtain Employee Details	Main	public void addEmployeeDetails(String employeeName, double salary)	This method is used to add the employee details into a map.
Obtain Employee Details	Main	public static EmployeeAudit findEmployee()	This method should return an EmployeeAudit object. To do this, implement the Lambda expression and identify the employees whose salary is less than or

			equal to the salary passed as an argument.

If the returned list is empty, then display "**No Employee Found**".

Use appropriate collections to perform the above tasks.

The signature of the above methods is given as part of the code skeleton, do not change the method signature.

In the Main class create a Main method with the menu as described in the sample Input and Output. When the user selects option **1. Add Employee details**, add the employeeName and salary to the **employeeMap**.

When the user selects option **2. Find Employee details**, it should display the employee name returned by the **findEmployee** method of **Main** class. If no employee is present, then it should display **" No Employee Found".**

When the user selects option **3. Exit**, display the message **"Let's complete the session"** and terminate the program.

Don't create an object for EmployeeAudit. Use the lambda expression.

In the Main class write the Main method and perform the given steps :

â€¢ Get the details of the Employees.

â€¢ Invoke the static method in the Main to identify whose salary is less than or equal to the salary specified

â€¢ Capture the object of EmployeeAudit returned by the static method.

â€¢ Display the result 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.

Ensure to provide the names for classes, attributes and methods as specified in the question.

Adhere to the code template, if provided.

Sample Input and Output:

```
1.Add Employee Details

2.Find Employee Details

3.Exit

 Enter the choice

 1

 Enter the Employee name

Faruq

Enter the Employee Salary

10000

1.Add Employee Details

2.Find Employee Details

3.Exit

 Enter the choice

1

Enter the Employee name

Benny

Enter the Employee Salary

20000

1.Add Employee Details

2.Find Employee Details

3.Exit

Enter the choice
```

```
2
Enter the salary to be searched

15000

Employee List

Faruq

1.Add Employee Details

2.Find Employee Details

3.Exit

Enter the choice

3

Let's complete the session
```

Travel Agency

Parveen agency, a registered IRCTC agent can book any number of e-Tickets. For Sleeper(SL) and Chair(2S), he earns a commission of Rs.60/passenger whereas, for First class AC(1A), Second class AC(2A) and Third class AC(3A) he earns Rs.100/passenger. The owner of Parveen agency decides to automate to calculate the commission he earned so far. Help him to calculate using Lambda expressions.

Requirement 1: Calculate the Commission amount

The owner wants to calculate the commission amount based on the class type.

Component Specification: Ticket(POJO class)

Type(class)	**Attributes**	**Methods**
Ticket	long pnrNo	Include the Getters and Setters. A lso write a 5 argument constructor in the order : pnrNo, passengerName, seatNo, classType and ticketFare

	String passengerName int seatNo String classType double ticketFare	

Note: classType is case in-sensitive

Component Specification: CommisionInfo Interface - This is a Functional Interface.

Type(Interface)	Methods	Responsibilities
CommisionInfo	public double calculateCommissionAmount(Ticket ticketObj)	This method is an abstract method used to calculate the amount he earns as his commission based on the class type using ticketObj.

Component Specification: UserInterfaceClass

Component Name	Type(Class)	Methods	Responsibilities
Generate the commission obtained	UserInterface	public static CommissionInfo generateCommissionObtained ()	This method should return a CommisionInfo object. To do this, implement the lambda expression to calculate the commission charges obtained per person based on the class type of the passenger.

The UserInterface class contains the main method.

In the main method,

- Get the passengers count and based on the count get the ticket details like: pnrNo, passengerName, seatNo, classType and ticketFare of each passenger.
- Create the Ticket object as an array.
- Invoke the static method generateCommissionObtained(). It returns a CommissionInfo object.
- Capture the CommissionInfo object in a reference variable.
- Using the reference of CommissionInfo, invoke the calculateCommissionAmount by passing the Ticket object as a parameter and capture the commission amount which is returned.
- The output should be displayed 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.
- Ensure to provide the name for classes, interfaces and methods as specified in the question.
- Adhere to the code template, if provided.
- Display the service charge correct to 2 decimal places. Use the System.out.printf method.

Sample Input 1:

Enter the no of passengers

5

Details of Passenger 1:

Enter the pnr no:

4617813567

Enter passenger name:

Arun

Enter seat no:

34

Enter class type:

1A

```
Enter ticket fare:

240

Details of Passenger 2:

Enter the pnr no:

4617813567

Enter passenger name:

Aruna

Enter seat no:

36

Enter class type:

2A

Enter ticket fare:

200

Details of Passenger 3:

Enter the pnr no:

4617813590

Enter passenger name:

Rachel

Enter seat no:

23

Enter class type:

2S

Enter ticket fare:
```

```
150

Details of Passenger 4:

Enter the pnr no:

4617813570

Enter passenger name:

Helen

Enter seat no:

48

Enter class type:

1a

Enter ticket fare:

240

Details of Passenger 5:

Enter the pnr no:

4617813567

Enter passenger name:

Andrews

Enter seat no:

78

Enter class type:

3a

Enter ticket fare:

240
```

Sample Output 1:

```
Commission Obtained

Commission obtained per each person: Rs.460.00
```

Eligibility Status

Practice editor

Validate Name

Create a public functional interface Validate with a method :

public boolean validateName(String name);

Create a **public class ValidateUtility** with the below methods :

public static Validate validateEmployeeName() - The lambda expression for the validateName method must return true if the name is valid and return false if the name is invalid.

In this case, the name is valid if it contains alphabets and space and it should contain minimum 5 characters and maximum 20 characters.

public static Validate validateProductName() - The lambda expression for the validateName method must return true if the name is valid and return false if the name is invalid.

In this case, the name is valid if the first character is an alphabet followed by 5 digits.

Write the main method in ValidateUtility class.

- Get the value for employee name and product name.

- Invoke the validateEmployeeName method

- Capture the object of Validate returned by the static method.

- Invoke the validateName method for the employee name received as input from the user.

- Display the result as shown in sample output.

- Next, invoke the validateProductName method

- Capture the object of Validate returned by the static method.

- Invoke the validateName method for the product name received as input from the user.

- Display the result as shown in sample output.

Note : Implement all the static methods mentioned above using lambda expression. Don't create object for the interface using new keyword.

Sample Input 1 :

Pinky Rose

A8546

Sample Output 1 :

Employee name is valid

Product name is invalid

Sample Input 2 :

Rahul@123

X82456

Sample Output 1 :

Employee name is invalid

Product name is valid

Mall Parking System

Mall Parking System

Westfield Shopping Mall is having a common two-wheeler parking stand in its basement. This parking facility is utilized by many of their customers regularly and they charge it for 10 Rupees per hour as parking fee. To make this facility more comfortable, they wish to automate the parking fee calculation based on the In-time and Out-time of a vehicle. For each vehicle entry, they will give a token which has the current date and time printed as In-time. The Out-time of each vehicle will the greater than the In-time. Help them to calculate the total Parking Fee by getting the In-time and Out-time as inputs.

Note:

- The input In-time, must be in dd/MM/yyyy HH:mm format and must be lesser than the current system time for at-least 1 Minute. Otherwise, print "<In-Time> is an Invalid In-Time" and terminate.
- The input Out-time, must be in dd/MM/yyyy HH:mm format and must be greater than the In-time for at-least 1 Minute. Otherwise, print "<Out-Time> is an Invalid Out-Time" and terminate.
- Output must be the calculated total parking fee based on difference of hours in between In-time and Out-time with 10 Rupees per hour as parking fee.

Assumption: Consider the current system date and time as 29/10/2019 20:10

Sample Input 1:

In-time

20/09/2019 23:55

Out-time

21/09/2019 23:56

Sample Output 1:

250 Rupees

Sample Input 2:

In-time

25/10/2019 03:05

Out-time

25/10/2019 03:06

Sample Output 2:

10 Rupees

Sample Input 3:

In-time

29/10/2019 20:10

Sample Output 3:

29/10/2019 20:10 is an Invalid In-Time

Explanation: Invalid In-time since it is not lesser than the current system date and time 29/10/2019 20:10 for at-least 1 Minute

Sample Input 4:

In-time

20/09/2019 23:55

Out-time

20/08/2019 23:55

Sample Output 4:

20/08/2019 23:55 is an Invalid Out-Time

Explanation: Invalid Out-time since it is not greater than the In-time for at-least 1 Minute

Employee Loan Eligibility

One of the major nationalized banks in the country offers a loan scheme based on the monthly salary to private sector employees. The minimum eligibility to avail this loan scheme is that the applicant must have a certain amount of salary as their minimum income per month. This minimum salary varies depending on the category of business. So, the bank is in need of an application to segregate the employee details. Help them by developing an application which takes all employee details as input and displays the shortlisted employee details as output.

You are provided with a code template which includes the following classes:

- Employee class which includes the attributes empId, empName and salary with all getters and setters.

- EmployeeUtility class which includes the **public Employee[] getEmployeeDetails (String[] details)** method, the **public Stream<Employee> getStreamOfEmployee (Employee[] empDetails)** method and the **public String[] shortlistedEmployee (Stream<Employee> empStream, Double minSalary**) method declarations.

- A main class with a main method which includes the entire user interface for this application.

Functional Requirements:

1. Enclose your code in the getEmployeeDetails method to convert the employee details from the array of Strings to the array of Employee objects.

2. Enclose your code in the getStreamOfEmployee method to convert the employee details from the array of Employee to the Stream of Employee.

3. Enclose your code in the shortlistedEmployee method to segregate the employee details that have at-least the minimumSalary value as their salary and return the sorted order of employee details based on empId as a String array.

User Interface Requirements:

- All the User Interface requirements will be provided in the code template itself. Adhere to the code template. Enclose your code in the respective required blocks of the EmployeeUtility class alone.

Note:

- The input format for each employee details is "**<empId;empName;salary>**"

- The Output format for each employee details is "**<empId> <empName> <salary>**"

- The empId may start with a numerical value or an alphabet as given in sample input and assume that all the empIds are unique. The output should be in ascending order based on the empId.

- Do not edit or delete the codes provided in the code template.

- Adhere to the Sample Inputs/ Outputs.

- In the Sample Inputs/ Outputs provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

Sample Input 1:

Enter the number of Employees

5

Enter the details of Employees

100;Adonis;25000

104;Andres;20000

103;Sai Shankar;35000

101;James;50000

102;Jack;30000

Enter the minimum eligible salary

30000

Sample Output 1:

Shortlisted Employees are

101 James 50000.0

102 Jack 30000.0

103 Sai Shankar 35000.0

Sample Input 2:

Enter the number of Employees

10

Enter the details of Employees

A100;Antony;27000

B104;Akbar;26000

100;Adonis;25000

104;Andres;17000

A103;Sophia;20000

103;Sai Shankar;35000

C101;Harry;20000

B102;Grace;30000

101;James;15000

102;Jack;30000

Enter the minimum eligible salary

25000

Sample Output 2:

Shortlisted Employees are

100 Adonis 25000.0

102 Jack 30000.0

103 Sai Shankar 35000.0

A100 Antony 27000.0

B102 Grace 30000.0

B104 Akbar 26000.0

Sample Input 3:

Enter the number of Employees

4

Enter the details of Employees

C101;Harry;20000

B102;Grace;30000

101;James;15000

102;Jack;30000

Enter the minimum eligible salary

35000

Sample Output 3:

No Employee is having the required salary

Placement Enrollment Count

A university organizes a placement drive in which students from all departments can participate. Each student has to register to this drive by submitting their roll number. The placement coordinator who organizes this event approaches you to develop an application to get the collection of registered student roll numbers which is separated by a comma and display the number of students based on the department.

All roll numbers are provided in a format in which the first two characters represent the department name through its acronym.

You are provided with a code template which includes the following class:

- Main class with main method declaration, **public static Stream<String> getRollNumbers(String rollNumbers)** method declaration and **public static int getCount(Stream<String> rollNumberStream, String deptAcronym)** method declaration.

Functional Requirements:

1. Enclose your code in the getRollNumbers method to separate each roll number from the bunch of rollNumbers provided with comma separated values. Convert the separated roll numbers into a Stream<String> and return the Stream.

2. Enclose your code in the getCount method to return the number of registered students from a particular department by comparing the rollNumberStream with deptAcronym (case insensitive).

3. Enclose your code in the main method to get the collection of comma separated roll numbers as a String and pass it to the getRollNumbers method to receive the Stream of roll numbers. Also, get the department name acronym as a string and pass it along with the received Stream of roll numbers to the getCount method to receive the number of students registered from that department.

User Interface Requirements:

- Display "Enter all roll numbers separated by comma" to get the collection of roll numbers.

- Display "Enter the department name acronym" to get the department name acronym.

- Display the number of registered students from the given department as "Number of students in <department name acronym> is <count>"

- If there is no roll numbers with the given department name acronym, then display "No students from <department name acronym>"

Note:

- The first input should be the roll numbers separated by a comma symbol (,) without any space in it.

- The second input is the department name acronym which is of two characters. (Case Insensitive)

- Assume that all roll numbers have seven characters where the first two are the department name acronym. The rest five are numerical digits.

- All inputs given for processing the functional requirements should be case insensitive.

- Adhere to the Sample Inputs/ Outputs

- In the Sample Inputs/ Outputs provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

Sample Input 1:

Enter all roll numbers separated by comma

ME12345,eC11452,EE21258,me12348,IT15445,Cs12345,EC11453,Ec22258,mE12348,IT15446,CS12345,ec11454,ee21258,Me12348,IT15447,cS12345,EC11455,EC21268,ME12348,ME15445,cs12357,ME12349,mE12350

Enter the department name acronym

me

Sample Output 1:

Number of students in me is 8

Sample Input 2:

Enter all roll numbers separated by comma

ME12345,eC11452,EE21258,me12348,IT15445,Cs12345,EC11453,Ec22258,mE12348,IT15446,CS12345,ec11454,ee21258,Me12348,IT15447,cS12345,EC11455,EC21268,ME12348,ME15445,cs12357,ME12349,mE12350

Enter the department name acronym

EC

Sample Output 2:

Number of students in EC is 6

Sample Input 3:

Enter all roll numbers separated by comma

ME12345,eC11452,EE21258,me12348,IT15445,Cs12345,EC11453,Ec22258,mE12348,IT15446,CS12345,ec11454,ee21258,Me12348,IT15447,cS12345,EC11455,EC21268,ME12348,ME15445,cs12357,ME12349,mE12350

Enter the department name acronym

AG

Sample Output 3:

No students from AG

Fruit Basket Estimation

A famous fruit stall in the marketplace approaches you to create an application in which their customers can estimate the total bill amount for the fruits in the basket. As a Java developer, create a Java application to add fruits to the basket and calculate the bill amount.

You are provided with a code template which includes the following classes:

- FruitBasket class which includes the attributes fruitName, weightInKgs and pricePerKg with all getters and setters, an empty constructor and a three argument constructor which is used to set values to all attributes.

- FruitBasketUtility class which includes the attribute fruitBasketList with its getter and setter, public void addToBasket(FruitBasket fbObj) method declaration and public int calculateBill(Stream<FruitBasket> fruitBasketStream) method declaration.

- Main class with a main method having user interface for getting the required inputs and pass them as an object to the addToBasket method.

Functional Requirements:

1. Enclose your code in the addToBasket method of the FruitBasketUtility class to add the FruitBasket object into the fruitBasketList.

2. Enclose your code in the calculateBill method to calculate the total bill amount from the Stream of FruitBasket objects. Each object will have a fruit detail. On multiplying the weightInKgs and pricePerKg of each FruitBasket object, the individual bill amount for that particular fruit can be calculated. This method should return the total bill amount by adding each individual bill amount of fruits present in the Stream.

3. Enclose your code in the Main class for retrieving the list of FruitBasket object from the FruitBasketUtility class and convert the List of objects into Stream of objects. Then pass the Stream of FruitBasket objects to the calculateBill method in FruitBasketUtility class and display the total bill amount.

User Interface:

- Display the options to choose as "Select an option: 1. Add Fruit to Basket 2. Calculate Bill 3.Exit" for all iterations.

- Any valid option can be chosen as 1 or 2 or 3. Otherwise, display a message as "Invalid option. Please try again." and continue to display the options.

- For option 1: get the fruitName, weightInKgs and pricePerKg as inputs to process the functional requirements and continue to display the options.

- For option 2: retrieve the list as mentioned in the functional requirements. If the retrieved list is empty, then display "**Your basket is empty. Please add fruits**." and continue to display the options. Otherwise, display the total bill amount as "**The estimated bill amount is Rs <total bill amount>"** and continue to display the options.

- For option 3: display a message "Thank you for using the application" and terminate.

Note:

- Few of the User Interface requirements will be provided in the code template itself. Adhere to the code template. Enclose your code in the respective required blocks alone.

- Do not edit or delete the codes provided in the code template.

- Adhere to the Sample Inputs/ Outputs.

- In the Sample Inputs/ Outputs provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

Sample Input/ Output 1:

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

2

Your basket is empty. Please add fruits.

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

1

Enter the fruit name

Mango

Enter weight in Kgs

5

Enter price per Kg

30

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

2

```
The estimated bill amount is Rs 150

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

1

Enter the fruit name

Apple

Enter weight in Kgs

2

Enter price per Kg

80

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

1

Enter the fruit name

Kiwi

Enter weight in Kgs

3

Enter price per Kg

45
```

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

2

The estimated bill amount is Rs 445

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

3

Thank you for using the application.

Sample Input/ Output 2:

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

4

Invalid option. Please try again.

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

2

Your basket is empty. Please add fruits.

Select an option:

1.Add Fruit to Basket

2.Calculate Bill

3.Exit

3

Thank you for using the application.

Employee Promotion

Neyon software limited has decided to give promotions to their employees based on their salary. The management has fixed a salary limit for the promotion. The management needs a report on the number of employees who have met the required condition.

Example: If the salary limit set by the management is 40000, then the employee whose salary is greater than or equal to 40000 should be considered for the count.

You being their software consultant have been approached to develop an application which can be used for managing their requirement. You need to implement a java program using multi-threading to find the count of employees as their requirement. Employee details should be obtained from the user on the console.

Component Specification: Employee (Model Class)

Type(Class)	**Attributes**	**Methods**

Employee	String empName String empId double empSalary	Include the getters and setter method for all the attributes. Include a three argument constructor in the given order - empName, empId and empSalary.

Note: The class and methods should be declared as public and all the attributes should be declared as private.

Requirement 1: Find the number of employees whose salaries are greater than or equal to the salary limit value set by the management.

Component Specification: Management (Thread Class)

Component Name	**Type(Class)**	**Attributes**	**Methods**
	Management	double salaryLimit List<Employee> employeeList int count	Include the getters and setters method for all th attributes. Include a tw argument Constructor t takes the arguments in t order salaryLimit, List<Employee>.
To **Find the count of** employees whose salaries are **greater than or equal to the salary limit value set by the management.**	**Management**		void run()

Note: The class and methods should be declared as public and all the attributes should be declared as private.

Create a class called Main with the main method and perform the given tasks:

- **Get the inputs like** the **number of employees,employee details, number of times to be searched and** the **salary limit from the user.**

The complex details will be in the form of a String in the following format

empName:empId:empSalary

- Parse the employee details and set the values for all the attributes in the **Employee** class using a **constructor**.
- Populate the Employee objects to the List.
- Get the number of times the salary limit value is to be entered by the user.
- Create a Management thread by passing the salary limit and the List created and then invoke the thread.
- Create a Management array and display the output 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.
Sample Input 1:
Enter the number of employees
4

Enter the employee details

Peter:E6322:65000

Alan:E6323:35000

Jeva:E6324:87000

Sam:E6325:40000

Enter the number of times salary limit to be searched

3

Enter the salary limit to be searched

40000

Enter the salary limit to be searched

75000

Enter the salary limit to be searched

30000

Sample Output 1:

40000.0 : 3

75000.0 : 1

30000.0 : 4

Add Flight using JDBC

Zaro Flight System wants to automate the process in their organization. As a start up, they need to automate the flight management system. Help them to develop this application.

You are provided with a public class Flight with following private attribute :

int flightId

String source

String destination

int noOfSeats

double flightFare

Appropriate setter and getter are written.

A public 5 argument constructor with arguments - flightId, source, destination, noOfSeats and flightFare is also provided.

Create a class FlightManagementSystem which has the following method. Use Database for manipulation.

public boolean addFlight(Flight flightObj) - This method should accept a flight object and add that flight details into the database. If flight details are added successfully, return true. Else, return false.

To connect to the database you are provided with database.properties file and DB.java file.

The flight table is already created at the backend. The structure of flight table is:

Column Name	Datatype
flightId	int
source	varchar2(30)
destination	varchar2(30)
noofseats	int
flightfare	number(8,2)

Create a class Main which has main method to perform the above operation.

In main method,

When **addFlight** method is invoked and if added successfully, print "Flight details added successfully" else print "Addition not done".

To execute on your machine, you can make the necessary changes to the values of connection url, username and password in the database.properties file.

Search for Trains - JDBC

Indian Railways Department wants to automate a process in their system. The train details are available in the database. The user should have a facility to search and view the trains based on a particular coach type available in the train which starts from a particular source place and ends in a particular destination place. A train can have various coach types among AC1, AC2, AC3, Sleeper and Seater.

You being their software consultant have been approached by them to develop an application which can be used for managing their business. You need to implement a Java program to view all the train which runs between a source and a destination with a particular coach type. Display the trains in a sorted order based on train number.

Component Specification: Train (Model Class)

Type(Class)	**Attributes**	**Methods**
Train	int trainNumber String trainName String source String destination int ac1 int ac2 int ac3 int sleeper int seater	

Note: Appropriate public getters,setters and a nine argument constructor in the given order - **trainNumber, trainName, source,destination, ac1,ac2,ac3,sleeper,seater** are provided as a part of the code template**.**

Requirement: Retrieve all the trains between source and destination with the required coach type

The user should have a facility to search and view the trains which runs between a source and a destination with a particular coach type. Hence the system should fetch all the train details for the given source and destination which contains the given coach type, from the database.

There can be more than one train which suits for the given requirement. So, the train details should be retrieved in a sorted order based on train number. Add all those train details into an ArrayList and return the same.

Component Specification: TrainManagementSystem

Component Name	Type (Class)	Methods	Responsibilities
Retrieve all the trains between source and destination with the given coachType	TrainManagementSystem	public ArrayList <Train> viewTrain (String coachType, String source, String destination)	This method should accept a coachType, source and destination as parameters and retrieve all the trains with the given coach type that runs between the source and destination from the database in ascending order based on train number. Return these details as ArrayList<Train>.

The **train** table is already created at the backend. The structure of train table is:

Column Name	Data Type
train_number	int
train_name	varchar(30)
source	varchar(30)
destination	varchar(30)
ac1	int
ac2	int
ac3	int
sleeper	int
seater	int

Sample records available in **train** table are:

train_number	train_name	source	destination	ac1	ac2	ac3	sleeper	seater
13005	Dehradun Mail	Howrah	Dehradun	3	0	3	5	0
13009	Doon Express	Howrah	Dehradun	5	5	10	0	0
22119	Pune Express	Mumbai	Pune	3	0	0	10	5
12123	Deccan Queen	Mumbai	Pune	0	7	0	5	0
15635	Dwarka Express	Okha	Guwahati	0	0	0	11	6
51422	Mumbai Pune Passenger	Mumbai	Pune	0	0	0	0	21
14645	Shalimar Express	Delhi	Jammu Tawi	0	8	7	0	2

To connect to the database you are provided with **database.properties** file and **DB.java** file. **(Do not change any values in database.properties file)**

Create a class called **Main** with the main method and get the inputs like **source, destination** and **coachType** from the user. The coachType should be any one among AC1, AC2, AC3, Sleeper and Seater (Case In-sensitive). If coachType is valid, then call **viewTrain** method in **TrainManagementSystem** class. Otherwise, display the output as "**Invalid Coach Type**".

Display the details of train such as trainNumber and trainName for all the trains returned as ArrayList<Train> from the method **viewTrain** in **TrainManagementSystem** class.

If no train is available as per the requirement, the output should be "**No trains found**".

Note:

- In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represents the output.

- Ensure to follow 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.

- In database, the **coach type** of **train** is given as separate columns. This represents the number of respective coaches available in the train.

- The following Sample Inputs / Outputs are generated based on the above mentioned Sample records given in **train** table.

Sample Input / Output 1:

Enter the source

Howrah

Enter the destination

Dehradun

Enter the coach type

AC2

13009 Doon Express

Sample Input / Output 2:

Enter the source

Mumbai

Enter the destination

Pune

Enter the coach type

Sleeper

12123 Deccan Queen

22119 Pune Express

Sample Input / Output 3:

Enter the source

Okha

Enter the destination

Guwahati

Enter the coach type

AC

Invalid Coach Type

Sample Input / Output 4:

Enter the source

Okha

Enter the destination

Guwahati

Enter the coach type

AC1

No trains found

Player Selection System_JDBC

The state government organizes a sports meet to encourage the school students. The selection committee for this event receives a huge number of registrations. So to organize things better they are in a need of an application to categorize the students registered as players based on their height (in centimetres) and weight (in Kilograms).

You being their software consultant have been approached to develop an application which can be used for managing their requirement. You need to implement a Java program to view the list of player's name based on their height and weight.

Component Specification: Player(Model Class)

Type(Class)	Attributes	Methods
Player	int playerId String playerName double height double weight	

Note: Appropriate public getters, setters and a four argument constructor in the given order - **playerId, playerName, height, weight** are provided as a part of the code template.

Requirement: Retrieve the list of player's name based on their height and weight.

The application should retrieve the player's names in ascending order, from the database based on the minimum height and maximum weight requirements. The shortlisted players should be within the range of the specified minimum height and the specified maximum weight (both inclusive).

Component Specification: PlayerSelectionSystem

Type (Class)	Methods	Responsibilities
PlayerSelectionSystem	public List<String> playersBasedOnHeightWeight (double minHeight, double maxWeight)	This method should accept minHeight and maxWeight as parameters and retrieve the pla names who are within the rang minHeight and maxWeight (bo inclusive) from the database. F these player's names in ascend order as a List of String.

The player table is already created in the backend. The structure of the table is as shown below:

Column Name	Data Type
playerId	int
playerName	varchar(30)
height	double(6,2)
weight	double(6,2)

Sample records available in the player table are as shown below:

playerId	playerName	height	weight
7507	Savannah	172	66
4588	Julian	152	50
324	Jacob	115	47

45	Thomas	164	65
7542	Charles	107	49
565	Maya	125	50
2458	Natalie	140	59
1237	Daisy	150	59

To connect to the database you are provided with the database.properties file and the DB.java file. (Do not change any values in the database.properties file)

Create a class called **Main** with the main method and get the inputs like the **minimum height** and the **maximum weight** from the user**.**

Display the player's names returned as List<String> from the method **playersBasedOnHeightWeight** in the **PlayerSelectionSystem** class.

If no player is available in the list, the output should be **"No players are with minimum height of <minHeight> and maximum weight of <maxWeight>"**

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.
- The following Sample Inputs / Outputs are generated based on the above mentioned Sample records are given in the player table.

Sample Input 1:

Enter the minimum height

140

Enter the maximum weight

65

Sample Output 1:

Daisy

Julian

Natalie

Thomas

Sample Input 2:

Enter the minimum height

120

Enter the maximum weight

35

Sample Output 2:

No players are with minimum height of 120.0 and maximum weight of 35.0

Sample Input 3:

Enter the minimum height

250

Enter the maximum weight

65

Sample Output 3:

No players are with minimum height of 250.0 and maximum weight of 65.0

Solutions:

Git Link:

https://github.com/thesaravanakumar/Cognizant-Early-Engagement/tree/main/Java%20Programming%20Funcamentals