**Day 13 Criteria**

Write a program to generate the below series:  
  
5,17,37,65,145, 197,….  


**Input Format:**

Input consists of a single integer which corresponds to n.

**Output Format:**

Output consists of the terms in the series separated by a blank space.

**Sample Input 1:**

6

**Sample Output 1:**

5 17 37 65 101 145

**Sample Input 2:**

15

**Sample Output 2:**

5 17 37 65 101 145 197 257 325 401 485 577 677 785 901

|  |
| --- |
| **import java.util.Scanner;**  **public class Main {**  **public static void main(String[] args) {**  **Scanner sc = new Scanner(System.in);**  **int n = sc.nextInt();**  **int a=2;**  **for(int i=1; i<=n;i++){**  **System.out.print(((a\*a)+1) +" ");**  **a=2\*(i+1);**  **}**    **}**  **}** |

**P3 - Palindrome**

A positive integer is called a palindrome if its representation in the decimal system is the same when read from left to right and from right to left.

Write a program to find whether the given positive integer K is a palindrome or not.

**Input Format**

The first line contains an integer, which corresponds to K.

**Output Format**

Output consists of a single string --- “palindrome” or “not a palindrome”.

**Sample Input 1:**

808

**Sample Output 1:**

palindrome

**Sample Input 2:**

2113

**Sample Output 2:**

not a palindrome

|  |
| --- |
| **import java.io.BufferedReader;**  **import java.io.InputStreamReader;**  **import java.text.SimpleDateFormat;**  **import java.util.\*;**  **public class Main {**  **public static void main(String[] args) {**  **//Fill your code here**  **Scanner sc = new Scanner(System.in);**  **int n=sc.nextInt();**  **int r,sum=0,temp=n;**  **while(n>0)**  **{**  **r=n%10;**  **sum=(sum\*10)+r;**  **n=n/10;**  **}**  **if(temp==sum)**  **{**  **System.out.println("palindrome");**  **}**  **else**  **{**  **System.out.println("not a palindrome");**  **}**  **}**  **}** |

**Alphabet Pattern 9**

Write a program to print the given pattern.

**Input Format:**

Input consists of a single integer which corresponds to the number of rows..

**Output Format:**

Refer sample output.

**Sample Input:**

5

**Sample Output:**

A

BB

CCC

DDDD

EEEEE

|  |
| --- |
| **import java.util.Scanner;**  **public class Main {**  **public static void main(String[] args) {**  **Scanner scan=new Scanner(System.in);**  **char c = 'A';**  **int rows =scan.nextInt();**  **for (int i = 0; i< rows; i++) {**  **for (int j = 0; j <= i; j++) {**  **System.out.print((char)(c+i));**  **}**  **System.out.println();**  **}**  **scan.close();**    **}**  **}** |

**P1 - Armstrong Number**

An Armstrong number of three digits is an integer such that the sum of the cubes of its digits is equal to the number itself. For example, 371 is an Armstrong number since 3^3 + 7^3 + 1^3 = 371.

Write a program to find whether a given 3-digit number is an Armstrong number or not.

**Input Format:**

Input consists of a single integer.

**Output Format:**

Refer sample output for details.

**Sample Input 1:**

153

**Sample Output 1:**

Armstrong Number

**Sample Input 2:**

101

|  |
| --- |
| **import java.io.BufferedReader;**  **import java.io.InputStreamReader;**  **import java.text.SimpleDateFormat;**  **import java.util.\*;**  **public class Main {**  **public static void main(String[] args) {**  **int num,number,temp,total=0;**  **Scanner sc=new Scanner(System.in);**  **num=sc.nextInt();**  **number=num;**  **for(;number!=0;number/=10)**  **{**  **temp=number%10;**  **total=total+temp\*temp\*temp;**  **}**  **if(total==num)**  **System.out.println("Armstrong Number");**  **else**  **System.out.println("Not An Armstrong Number");**  **}**  **}** |

**P3 – Number series**

Write a program to print the series ---- 1,3,6,10,15 ….. upto ‘n’ terms.

**Input Format:**

Input consists of a single integer.

**Output Format:**

Refer sample output for details.

**Sample Input:**

6

**Sample Output:**

1 3 6 10 15 21

|  |
| --- |
| **import java.util.Scanner;**  **public class Main {**  **public static void main(String[] args) {**  **// TODO Auto-generated method stub**  **Scanner sc=new Scanner(System.in);**  **int terms=sc.nextInt();**  **int n=1;**  **int m=2;**  **System.out.print(n+" ");**  **for(int i=1;i<terms;i++)**    **{**  **n=n+m;**  **System.out.print(n+" ");**  **m++;**  **}**  **}**  **}** |

**Pattern 1**

Write a program to print the given pattern.

**Input Format:**

Input consists of a single integer.

**Output Format:**

Refer sample outputs. There is a trailing space at the end of each line.

**Sample Input 1:**

5

**Sample Output 1:**

1 2 3 4 5

1 2 3 4

1 2 3

1 2

1

**Sample Input 2:**

3

**Sample Output 2:**

1 2 3

1 2

1

|  |
| --- |
| **import java.util.Scanner;**  **public class Main {**  **public static void main(String[] args) {**  **Scanner sc=new Scanner(System.in);**  **int rows=sc.nextInt();**  **int i,j;**  **for (i=rows; i>=1;i--){**  **for (j=1; j<=i; j++){**  **System.out.print(j+" ");**  **}**    **System.out.println();**  **}**  **}**  **}** |

**Lucky Customer Award**

Every day few of the customers are given a lucky gift. Lucky gift is given to a customer when his / her bill number ends with the last digit of that day number or when the bill number is a multiple of the day number.Can you help Gita in deciding whether a customer gets the lucky gift or not?

**Input Format:**

Input consists of 2 integers that correspond to the day number in today's date and the bill number.  
  
**Output Format:**

Output is either 'yes' or 'no'. Output is yes when the customer gets the lucky gift and is no otherwise.

**Sample Input 1:**

5

45

**Sample Output 1:**

yes

**Sample Input 2:**

14

34

**Sample Output 2:**

yes

**Sample Input 3:**

5

44

**Sample Output 3:**

no

|  |
| --- |
| **import java.io.\*;**  **import java.text.\*;**  **import java.util.\*;**  **public class Main**  **{**    **public static void main(String[] args) throws IOException**  **{**  **Scanner scan=new Scanner(System.in);**  **int num1 = scan.nextInt();**  **int num2 = scan.nextInt();**    **if(num1%10==num2%10|| num2%num1==0){**  **System.out.println("yes");**  **}**  **else{**  **System.out.println("no");**  **}**  **}**  **}** |

**Alphabet Pattern 1**

Write a program to print the given pattern.

**Input and Output Format:**

Input consists of a single integer that corresponds to the number of rows,n.

The output is the alphabet pattern for the given input,n.

**Sample Input 1:**

5

**Sample Output 1:**

A

AB

ABC

ABCD

ABCDE

**Sample Input 2:**  
7

**Sample Output 2:**  
A  
AB  
ABC  
ABCD  
ABCDE  
ABCDEF  
ABCDEFG

|  |
| --- |
| **import java.util.\*;**  **public class Main {**  **public static void main(String[] args) {**  **int i,j;**  **Scanner sc = new Scanner(System.in);**  **int n=sc.nextInt();**    **for(i=1;i<=n;i++)**  **{**  **for(j=1;j<=i;j++)**  **{**  **System.out.print((char)(j+64));**  **}**  **System.out.println("");**  **}**  **}**  **}** |

**Day 11 Criteria**

Write a program to generate the below series:

24,60,120,210,…

**Input Format:**

Input consists of a single integer that corresponds to n.

**Output Format:**

The output consists of the terms in the series separated by a blank space.

**Sample Input 1:**

5

**Sample Output 1:**

24 60 120 210 336

**Sample Input 2:**

10

**Sample Output 2:**

24 60 120 210 336 504 720 990 1320 1716

|  |
| --- |
| **import java.util.\*;**  **public class Main {**  **public static void main(String[] args) {**    **Scanner sc=new Scanner(System.in);**  **int n=sc.nextInt();**  **for(int i=2;i<=n+1;i++)**  **{**  **int j=i+1;**  **int k=i+2;**  **System.out.print(i\*j\*k+ " ");**  **j=0;**  **k=0;**  **}**  **}**  **}** |

**Alphabet Pattern 4**

Write a program to print the given pattern.

**Input Format:**

Input consists of a single integer which corresponds to the number of rows..

**Output Format:**

Refer sample output.

**Sample Input:**

5

**Sample Output:**

E

ED

EDC

EDCB

EDCBA

|  |
| --- |
| **import java.util.\*;**  **public class Main {**  **public static void main(String[] args) {**  **int i,j,n;**  **Scanner sc = new Scanner(System.in);**  **n = sc.nextInt();**  **for( i=n;i>=1;i--)**  **{**  **for( j=n;j>=i;j--)**  **{**  **System.out.print((char)(j+64));**  **}**  **System.out.println();**  **}**  **}**  **}** |

**Pattern 3**

Write a program to print the given pattern.

**Input Format:**

Input consists of a single integer.

**Output Format:**

Refer sample outputs. There is a trailing space at the end of each line.

**Sample Input:**

5

**Sample Output:**

5 4 3 2 1  
4 3 2 1  
3 2 1  
2 1  
1

|  |
| --- |
| **import java.util.\*;**  **public class Main {**  **public static void main(String[] args) {**  **Scanner sc = new Scanner(System.in);**  **int n = sc.nextInt();**  **for(int x=n;x>=1;x--)**  **{**  **for (int y=x;y>0;y--)**  **System.out.print(y + " ");**  **System.out.println();**  **}**  **}**  **}** |

**Palindromic Prize**

A customer in the Personalised Gift Store is awarded a prize when their bill number is a 3-digit palindrome.

Write a program for identifying the prize winners.

**Input Format:**

Input consists of a number that corresponds to the bill number.

**Output Format:**

The output consists of a string that is either 'yes' or 'no'. The output is 'yes' when the customer receives the prize and is 'no' otherwise.

**Sample Input 1:**

565

**Sample Output 1:**

yes

**Sample Input 2:**

568

**Sample Output 2:**

no

**Sample Input 3:**

66

**Sample Output 3:**

no

|  |
| --- |
| **import java.util.Scanner;**  **public class Main {**  **public static void main(String[] args) {**  **Scanner scan = new Scanner(System.in);**  **int billNo = scan.nextInt();**  **if((billNo/100)==(billNo%10))**  **System.out.println("yes");**  **else**  **System.out.println("no");**  **scan.close();**  **}**  **}** |

**GRADE**

Write a program to determine the grade of the student in a particular subject. Refer to the table given below for grade details.

|  |  |
| --- | --- |
| **Marks scored** | **Grade** |
| 100 | S |
| [90,100) | A |
| [80,90) | B |
| [70,80) | C |
| [60,70) | D |
| [50,60) | E |
| <50 | F |

The interval **[a,b)** includes all numbers greater than or equal to **a** and less than **b**.

**Input and Output Format:**

Input consists of a single integer that corresponds to the marks scored by the student.

Print **"Invalid Input"** if it is not in the range **0 to 100**.

Refer sample input and output for formatting specifications.

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

**Sample Input and Output 1:**

Enter the marks

**85**

The student obtained a B grade

**Sample Input and Output 2:**

Enter the marks

**850**

Invalid Input

OBSERVATION

2

|  |
| --- |
| **import java.util.Scanner;**  **public class Main {**  **public static void main(String[] args) {**  **Scanner sc = new Scanner(System.in);**  **System.out.println("Enter the marks");**  **double Marks = sc.nextDouble();**  **if(Marks == 100){**  **System.out.println("The student obtained a S grade");**  **}**  **else if(Marks<100 && Marks>= 90){**  **System.out.println("The student obtained a A grade");**    **}**  **else if(Marks<90 && Marks>= 80){**  **System.out.println("The student obtained a B grade");**  **}**  **else if(Marks<80 && Marks>=70){**  **System.out.println("The student obtained a C grade");**  **}**  **else if(Marks<70 && Marks>=60){**  **System.out.println("The student obtained a D grade");**  **}**  **else if(Marks<60 && Marks >=50){**  **System.out.println("The student obtained a E grade");**    **}**  **else if (Marks < 50 && Marks >=0){**  **System.out.println("The student obtained a F grade");**    **}**  **else**  **{**  **System.out.println("Invalid Input");**  **}**  **//fill your code here**  **}**  **}** |

**Abstract class and methods**

Write a Java program to display the package details along with the cost of package per month using abstract class and methods.

**Strictly adhere to the Object Oriented Specifications given in the problem statement. All class names, member variable names and function names should be the same as specified in the problem statement.**

Create an abstract class named **Package**with the following protected attributes.

|  |  |
| --- | --- |
| **Data Type** | **Attributes** |
| String | name |
| Boolean | isAvailable |
| String | city |
| Integer | rentalPeriod |
| Double | totalCost |

Include appropriate **getters**and **setters**and **constructors**for the above class.  
Include the abstract method for **display()**for the abstract class.

The class **Package**should have the following non abstract method.

|  |  |
| --- | --- |
| **Method name** | **Description** |
| Double calculatePackageCost() | This method is used to calculate the cost of package per month by dividing the total cost with the rental period and return the same. |

Create a class named **CustomPackage**which extends**Package**with the following private attributes.

|  |  |
| --- | --- |
| **Data Type** | **Data Member** |
| String | productType |
| Integer | numberOfProducts |

Create **default constructor** and a **parameterized constructor** with arguments in order CustomPackage (String name, Boolean isAvailable, String city, Integer rentalPeriod, Double totalCost, String productType, Integer numberOfProducts). Include appropriate **getters**and **setters**.

The class **CustomPackage**should implement the following method

|  |  |
| --- | --- |
| **Method name** | **Description** |
| void display() | This method is used to display all the details of the package. Also it should display the total cost and cost of package/month at the end with two decimal places. |

Create a class named **LivingRoomPackage**which extends **Package**with the following private attributes.

|  |  |
| --- | --- |
| **Data Type** | **Data Member** |
| Integer | noOfFan |
| Integer | noOfChair |
| Integer | noOfWadrobe |

Create **default constructor** and a **parameterized constructor** with arguments in order LivingRoomPackage (String name, Boolean isAvailable, String city, Integer rentalPeriod, Double totalCost, Integer noOfFan, Integer noOfChair, Integer noOfWadrobe). Include appropriate **getters**and **setters**.

The class **LivingRoomPackage**should implement the following method

|  |  |
| --- | --- |
| **Method name** | **Description** |
| void display() | This method is used to display all the details of the package. Also it should display the total cost and cost of package/month at the end with two decimal places. |

Create a driver class called  **Main** . In the Main method, obtain input from the user in CSV format and split the input using split method.  
  
**Note:** The products or packages in instasmart are rented based on the month, hence the total cost of the package has to be divided by the number of months(rental period) requested by the customer.  
  
**Input and Output format:**  
Refer sample Input and Output for formatting specifications.  
   
**[All text in bold corresponds to input and the rest corresponds to output]**  
**Sample Input and Output 1:**  
1.Custom Package  
2.Living Room Package  
Enter your choice  
**1**  
Enter the custom package details in comma separated format:  
**Package-1,1,Chennai,4,1500.55,Steel,3**  
Custom Package details:  
Package name : Package-1  
Availability : true  
City : Chennai  
Rental period : 4  
Product type : Steel  
No. of products : 3  
Total Cost : 1500.55  
Cost of package/month : 375.14  
  
**Sample Input and Output 2:**  
1.Custom Package  
2.Living Room Package  
Enter your choice  
**2**  
Enter the living room package details in comma separated format:  
**Package-1,0,Chennai,3,1500.55,4,2,3**  
Living Room Package details:  
Package name : Package-1  
Availability : false  
City : Chennai  
Rental period : 3  
No. of Fan : 4  
No. of Chair : 2  
No. of Wardrobe : 3  
Total Cost : 1500.55  
Cost of package/month : 500.18  
  
**Sample Input and Output 3:**  
1.Custom Package  
2.Living Room Package  
Enter your choice  
**3**  
Invalid choice

|  |  |  |  |
| --- | --- | --- | --- |
| public abstract class Package {  protected String name;  protected Boolean isAvailable;  protected String city;  protected int rentalPeriod;  protected double totalCost;  public Package(String name, Boolean isAvailable, String city, int rentalPeriod, double totalCost) {  super();  this.name = name;  this.isAvailable = isAvailable;  this.city = city;  this.rentalPeriod = rentalPeriod;  this.totalCost = totalCost;  }  public Package() {}    public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public Boolean getIsAvailable() {  return isAvailable;  }  public void setIsAvailable(Boolean isAvailable) {  this.isAvailable = isAvailable;  }  public String getCity() {  return city;  }  public void setCity(String city) {  this.city = city;  }  public int getRentalPeriod() {  return rentalPeriod;  }  public void setRentalPeriod(int rentalPeriod) {  this.rentalPeriod = rentalPeriod;  }  public double getTotalCost() {  return totalCost;  }  public void setTotalCost(double totalCost) {  this.totalCost = totalCost;  }  Double calculatePackageCost() {  return totalCost/rentalPeriod;  }  } | import java.text.\*;  public class LivingRoomPackage extends Package {    private int noOfFan;  private int noOfChair;  private int noOfWardrobe;  public LivingRoomPackage(String name, Boolean isAvailable, String city, int rentalPeriod, double totalCost,  int noOfFan, int noOfChair, int noOfWardrobe) {  super(name, isAvailable, city, rentalPeriod, totalCost);  this.noOfFan = noOfFan;  this.noOfChair = noOfChair;  this.noOfWardrobe = noOfWardrobe;  }  DecimalFormat df = new DecimalFormat("#0.00");  Package p = new Package() {};  public int getNoOfFan() {  return noOfFan;  }  public void setNoOfFan(int noOfFan) {  this.noOfFan = noOfFan;  }  public int getNoOfChair() {  return noOfChair;  }  public void setNoOfChair(int noOfChair) {  this.noOfChair = noOfChair;  }  public int getNoOfWardrobe() {  return noOfWardrobe;  }  public void setNoOfWardrobe(int noOfWardrobe) {  this.noOfWardrobe = noOfWardrobe;  }  void display() {  System.out.println("Living Room Package details:");  System.out.println("Package name : " + getName());  System.out.println("Availability : " + getIsAvailable());  System.out.println("City : " + getCity());  System.out.println("Rental period : " + getRentalPeriod());  System.out.println("No. of Fan : " + noOfFan);  System.out.println("No. of Chair : " + noOfChair);  System.out.println("No. of Wardrobe : " + noOfWardrobe);  System.out.println("Total Cost : " + df.format(getTotalCost()));  System.out.println("Cost of package/month : " + df.format(calculatePackageCost()));  //String.format("$%,.2f", calculatePackageCost());  }    } | import java.text.\*;  public class CustomPackage extends Package {  private String productType;  private Integer numberOfProducts;  public CustomPackage(String name, Boolean isAvailable, String city, int rentalPeriod, double totalCost,  String productType, Integer numberOfProducts) {  super(name, isAvailable, city, rentalPeriod, totalCost);  this.productType = productType;  this.numberOfProducts = numberOfProducts;  }  DecimalFormat df = new DecimalFormat("#0.00");  public String getProductType() {  return productType;  }  public void setProductType(String productType) {  this.productType = productType;  }  public Integer getNumberOfProducts() {  return numberOfProducts;  }  public void setNumberOfProducts(Integer numberOfProducts) {  this.numberOfProducts = numberOfProducts;  }  void display() {  System.out.println("Custom Package details:");  System.out.println("Package name : " + getName());  System.out.println("Availability : " + getIsAvailable());  System.out.println("City : " + getCity());  System.out.println("Rental period : " + getRentalPeriod());  System.out.println("Product type : " + productType);  System.out.println("No. of products : " + numberOfProducts);  System.out.println("Total Cost : " + df.format(getTotalCost()));  System.out.println("Cost of package/month : " + df.format(calculatePackageCost()));  }      } | import java.io.\*;  public class Main {  public static void main(String args[]) throws Exception {  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  System.out.println("1.Custom Package\n2.Living Room Package\nEnter your choice");  Integer choice = Integer.parseInt(br.readLine());  String input = "";  Boolean isAvailable = false;  switch(choice) {  case 1:  System.out.println("Enter the custom package details in comma separated format:");  input = br.readLine();  String str[] = input.split(",");  if(Integer.parseInt(str[1])==1)  isAvailable = true;  CustomPackage c = new CustomPackage(str[0],isAvailable,str[2],Integer.parseInt(str[3]),Double.parseDouble(str[4]),str[5],Integer.parseInt(str[6]));  c.display();  break;  case 2:  System.out.println("Enter the living room package details in comma separated format:");  input = br.readLine();  String st[] = input.split(",");  if(Integer.parseInt(st[1])==1)  isAvailable = true;  LivingRoomPackage l = new LivingRoomPackage(st[0],isAvailable,st[2],Integer.parseInt(st[3]),Double.parseDouble(st[4]),Integer.parseInt(st[5]),Integer.parseInt(st[6]),Integer.parseInt(st[7]));  l.display();  break;  default:  System.out.println("Invalid choice");  break;  }  }  } |

**Discounts - Inheritance**

One of the easier ways to identify the scenarios that reflect inheritance is to look for a "is-a" relationship in the requirements document. On trying to check if we have such hierarchies, we find that there are different types of customers/account holders in the Bank. Customers can be Normal, Priviledged, SeniorCitizen and so on.   
The Bank also introduces an offer where privileged customers get a 30% off on the bill while senior citizens get 12% off.  
  
Lets implement the inheritance for the given scenario yet again for a better understanding.  
  
1. Create Customer, Privileged & SeniorCitizen class with data members as given below.  
2. Implement generateBillAmount Method as per the specification.

Create a class **Customer** with the following private data members

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| String | name |
| String | address |
| Integer | age |
| String | mobileNumber |

Methods in class **Customer**

|  |  |
| --- | --- |
| **Method Name** | **Method description** |
| displayCustomer() | To display the details of the customer. |

Use Appropriate **Getters & Setters**for **Customer**class.

Create a class **SeniorCitizenCustomer** which extends the class **Customer**.  
Methods in class **SeniorCitizenCustomer**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Method description** | **Return Type** |
| generateBillAmount(amount) | To calculate the payment amount where the discount is 12% . | Double |

Create a class **PrivilegeCustomer** which extends the class **Customer**.  
Methods in class **PrivilegeCustomer**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Method description** | **Return Type** |
| generateBillAmount(amount) | To calculate the payment amount where the discount is 30% . | Double |

Create a driver class named **Main** which creates an instance of the above mentioned classes.  
Use setters to set the values to objects and display all details using getters from the main method.

**Note :**

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

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

**Input and Output Format:**

Refer sample input and output for formatting specifications.

**[All text in bold corresponds to input and the rest corresponds to output.]  
Sample Input and Output 1:**

1)Privilege Customer  
2)SeniorCitizen Customer  
Enter Customer Type  
**1**  
Enter The Name  
**Ram**  
Enter The Age  
**25**  
Enter The Address  
**CBE**  
Enter The Mobile Number  
**9576531641**  
Enter The Purchased Amount  
**5000**  
Bill Details  
Name Ram  
Mobile 9576531641  
Age 25  
Address CBE  
Your bill amount is Rs 5000.0. Your bill amount is discount under privilege customer  
You have to pay Rs 3500.00

**Sample Input and Output 2:**

1)Privilege Customer  
2)SeniorCitizen Customer  
Enter Customer Type  
**3**  
Invalid Customer Type

|  |  |  |  |
| --- | --- | --- | --- |
| **public class Customer {**  **private String name;**  **private String address;**  **private Integer age;**  **private String mobileNumber;**    **// fill your code here**  **void displayCustomer() {**  **//fill your code here**  **System.out.println("Bill Details");**  **System.out.println("Name "+ this.getName());**  **System.out.println("Mobile " + this.mobileNumber);**  **System.out.println("Age " +this.getAge());**  **System.out.println("Address " +this.getAddress());**  **}**  **public String getName() {**  **return name;**  **}**  **public void setName(String name) {**  **this.name = name;**  **}**  **public String getAddress() {**  **return address;**  **}**  **public void setAddress(String address) {**  **this.address = address;**  **}**  **public Integer getAge() {**  **return age;**  **}**  **public void setAge(Integer age) {**  **this.age = age;**  **}**  **public String getMobileNumber() {**  **return mobileNumber;**  **}**  **public void setMobileNumber(String mobileNumber) {**  **this.mobileNumber = mobileNumber;**  **}**      **}**  **public class PrivilegeCustomer extends Customer{**    **double generateBillAmount(Double amount) {**    **//fill your code here**  **double dis = (amount \* 30)/100;**    **return amount - dis;**    **}**    **}** | **public class PrivilegeCustomer extends Customer{**  **double generateBillAmount(Double amount) {**  **double dis = (amount \* 30)/100;**  **return amount - dis;**  **}**  **}** | **public class SeniorCitizenCustomer extends Customer{**    **double generateBillAmount(Double amount) {**  **double dis = (amount \* 12)/100;**  **return amount - dis;**  **}**  **}** | **import java.text.DecimalFormat;**  **import java.util.Scanner;**  **public class Main {**  **public static void main(String[] args) {**  **Scanner sc = new Scanner(System.in);**  **System.out.println("\n1)Privilege Customer\n2)SeniorCitizen Customer\nEnter Customer Type");**  **int ch = sc.nextInt();**  **DecimalFormat df = new DecimalFormat("0.00");**  **switch (ch) {**    **case 1 : PrivilegeCustomer pc = new PrivilegeCustomer();**  **sc.nextLine();**  **System.out.println("Enter The Name");**  **pc.setName(sc.nextLine());**  **System.out.println("Enter The Age");**  **pc.setAge(sc.nextInt());**  **sc.nextLine();**  **System.out.println("Enter The Address");**  **pc.setAddress(sc.nextLine());**  **System.out.println("Enter The Mobile Number");**  **pc.setMobileNumber(sc.nextLine());**  **System.out.println("Enter The Purchased Amount");**  **double amount = sc.nextDouble();**    **double total = pc.generateBillAmount(amount);**  **pc.displayCustomer();**  **System.out.println("Your bill amount is Rs "+amount+". Your bill amount is discount under privilege customer");**  **System.out.println("You have to pay Rs "+df.format(total));**  **break;**  **case 2 :**  **SeniorCitizenCustomer scc = new SeniorCitizenCustomer();**  **sc.nextLine();**  **System.out.println("Enter The Name");**  **scc.setName(sc.nextLine());**  **System.out.println("Enter The Age");**  **scc.setAge(sc.nextInt());**  **sc.nextLine();**  **System.out.println("Enter The Address");**  **scc.setAddress(sc.nextLine());**  **System.out.println("Enter The Mobile Number");**  **scc.setMobileNumber(sc.nextLine());**  **System.out.println("Enter The Purchased Amount");**  **double amount1 = sc.nextDouble();**    **double total1 = scc.generateBillAmount(amount1);**  **scc.displayCustomer();**  **System.out.println("Your bill amount is Rs "+amount1+". Your bill amount is discount under senior citizen customer");**  **System.out.println("You have to pay Rs "+df.format(total1));**  **break;**  **default : System.out.println("Invalid Customer Type");**  **break;**  **}**    **}**  **}** |

**Call History**

Practice makes a man perfect !!! Let us use FileWriter & BufferedWriter to read call log data from console and rewrite it into a CSV file. Please refer to the specification given below.  
**Note :**Read the input from the user and write using 'FileWriter writer = new FileWriter("call.csv");'  
Name the output file as **'call.csv'.**  
**Sample Input and Output:**  
**[All text in bold are input and the remaining are output]**

Enter the mobile number

**8972007627**

Enter the duration (in Seconds)

**5**

Do you want to add another call history ?

**yes**

Enter the mobile number

**8976543289**

Enter the duration (in Seconds)

**6**

Do you want to add another call history ?

**no**

**Output(call.csv):**

8972007627,5

8976543289,6

|  |
| --- |
| **import java.io.FileWriter;**  **import java.io.IOException;**  **import java.util.Scanner;**  **public class Main {**  **public static void main(String[] args) {**  **//fill your code here**    **Scanner sc = new Scanner(System.in);**  **String a;**  **try {**  **FileWriter w = new FileWriter("call.csv");**  **do {**  **System.out.println("Enter the mobile number");**  **String num = sc.nextLine();**  **w.append(num);**  **w.append(",");**  **System.out.println("Enter the duration (in Seconds)");**  **String sec = sc.nextLine();**  **w.append(sec);**  **w.append("\n");**  **System.out.println("Do you want to add another call history ?");**  **a = sc.nextLine();**  **w.flush();**  **}while(a.equals("yes"));**  **w.close();**  **}catch (IOException e) {**  **System.out.println(e);**  **}**      **}**  **}** |

**SDF Exception**

Lets' try to catch a very specific exception, i.e, SimpleDateFormat Exception, which is thrown when we read a String and try to convert it into Date. Assume we are reading Date-of-Birth of a customer and if they enter it in an incorrect format, capture it & display a message.

Write a program to get the customer account name, account type and Date-Of-Birth. Validate the Date-Of-Birth details and display the details.  
  
Create a class**Account** with following attributes

|  |  |
| --- | --- |
| **Data type** | **Variable name** |
| String | accountName |
| String | accountType |
| Date | dob |

Include a **default**and **parameterized constructor**with all the variables.

Include the following methods in the class **Account**.

|  |  |
| --- | --- |
| **Method name** | **Method description** |
| void display() | This method is used to display the details |
| boolean validateDOB(String dob) | This method is used to validate the given date of birth and return true or false based on the validation. |

**Sample Input and Output:**  
**[All text in bold corresponds to input and the rest corresponds to output]**  
Enter the Customer details  
Enter the name  
**Sastha**  
Enter account type  
**Savings**  
Enter date-of-birth  
**02/13/1993**  
Wrong Format(eg:01/01/2015)  
Enter date-of-birth  
**35/02/1993**  
Wrong Format(eg:01/01/2015)  
Enter date-of-birth  
**25/02/1993**  
  
Account Details  
Name : Sastha  
Type : Savings  
D.O.B : Feb-25-1993

|  |  |
| --- | --- |
| **import java.util.Date;**  **import java.text.DateFormat;**  **import java.text.ParseException;**  **import java.text.SimpleDateFormat;**  **public class Account {**  **String accountName;**  **String accountType;**  **Date dob;**  **public String getAccountName() {**  **return accountName;**  **}**  **public void setAccountName(String accountName) {**  **this.accountName = accountName;**  **}**  **public String getAccountType() {**  **return accountType;**  **}**  **public void setAccountType(String accountType) {**  **this.accountType = accountType;**  **}**  **public Date getDob() {**  **return dob;**  **}**  **public void setDob(Date dob) {**  **this.dob = dob;**  **}**  **public Account() {**  **super();**  **// TODO Auto-generated constructor stub**  **}**  **public Account(String accountName, String accountType, Date dob) {**  **super();**  **this.accountName = accountName;**  **this.accountType = accountType;**  **this.dob = dob;**  **}**  **void display() throws ParseException{**    **System.out.println("\nAccount Details");**  **System.out.println("Name : " + getAccountName());**  **System.out.println("Type : " + getAccountType());**    **SimpleDateFormat outputbirthformat = new SimpleDateFormat("MMM-dd-yyyy");**    **String dob = outputbirthformat.format(getDob());**  **System.out.println("D.O.B : " + dob);**    **}**    **boolean validateDOB(String dob) {**  **boolean result = false;**  **String[] numbers = dob.split("/");**  **int day = Integer.parseInt(numbers[0]);**  **int month = Integer.parseInt(numbers[1]);**    **try {**    **int[] numberOfDaysEachMonth = new int[] {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};**  **if (month > 0 && month < 13) {**  **if (day > 0 && day < numberOfDaysEachMonth[month - 1]) {**  **result = true;**  **} else {**  **System.out.println("Wrong Format(eg:01/01/2015)");**    **result = false;**  **}**  **} else {**  **System.out.println("Wrong Format(eg:01/01/2015)");**    **result = false;**  **}**  **} catch (Exception e) {**  **//System.out.println(e);**  **}**  **return result;**  **}**  **}** | **import java.text.DateFormat;**  **import java.text.ParseException;**  **import java.text.SimpleDateFormat;**  **import java.util.Scanner;**  **import java.util.Date;**  **public class Main {**  **public static void main(String[] args) throws ParseException{**    **Scanner sc = new Scanner(System.in);**  **SimpleDateFormat inputbirthformat = new SimpleDateFormat("dd/MM/yyyy");**    **System.out.println("Enter the Customer details");**  **System.out.println("Enter the name");**    **String name = sc.next();**    **System.out.println("Enter account type");**  **String accType = sc.next();**    **Account ac = new Account();**  **boolean status = false;**  **String dob;**    **do{**  **System.out.println("Enter date-of-birth");**  **dob = sc.next();**      **status = ac.validateDOB(dob);**    **}while(status != true);**    **Date date1 = inputbirthformat.parse(dob);**  **Account pac = new Account(name,accType,date1);**  **pac.display();**  **// Date date2 = outputbirthformat.parse(dob);**  **// System.out.println("D.O.B : " + date2);**      **}**  **}** |

**Interface Practical Problem 2**

All the banks will have their own rules to be followed but there are few areas where every bank has to follow the RBI rules. So when RBI changes any condition it must be followed by all other banks.  
  
Lets think that RBI comes and says that there will be a fixed credit score . So every bank must follow that credit score for all the transactions. In our problem let us consider the fixed credit score to be 10%. So we have to calculate the credit score for the customer.  
  
Create a interface **Bank**with method - **calculateCreditScore()** of return type as double.  
Create a class **RBI** which implemets the interface Banks with 3 private data member variables -**accountNumber** of type String, **creditScore** of type double , **holderName** of type String and a fixed variable **CREDIT** of type double. Include the method **calculateCreditScore()** and **display()**.  
   
Create the class **ICICI** which extends the class **RBI** .  
  
Create the class **HDFC** which extends the class **RBI** .  
  
Use Appropriate Getters Setters for above classes.  
  
Create a driver class named Main which creates an instance of the above mentioned classes. Credit score must be calculated seperately for all the classes  (value must be round to 2 decimal place).  
  
**[All text in bold corresponds to input and the rest corresponds to output.]**  
  
**Sample Input and Output 1:**  
  
Select the Bank Name  
1.ICICI  
2.HDFC  
**1**  
Enter the Holder Name  
**Madhan Kumar**  
Enter the Account Number  
**218463**  
Enter the Previous Credit Score  
**652**  
Enter the Amount to be Paid  
**500**  
Amount Paid Successfully !!!  
Hi,Madhan Kumar  
You have gained 50.00 credit score for the payment of 500.0  
Your Total Credit Score is 702.00  
  
**Sample Input and Output 2:**  
  
Select the Bank Name  
1.ICICI  
2.HDFC  
**2**  
Enter the Holder Name  
**Raina**  
Enter the Account Number  
**62354953**  
Enter the Previous Credit Score  
**6015**  
Enter the Amount to be Paid  
**15600**  
Amount Paid Successfully !!!  
Hi,Raina  
You have gained 1560.00 credit score for the payment of 15600.0  
Your Total Credit Score is 7575.00  
  
**Sample Input and Output 3:**  
  
Select the Bank Name  
1.ICICI  
2.HDFC  
**6**  
Invalid Bank type

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **public interface Bank {**  **abstract double calculateCreditScore();**  **}** | **import java.text.DecimalFormat;**  **public class HDFC extends RBI {**  **private String accountNumber;**  **private double creditScore;**  **private String holderName;**  **static double CREDIT;**  **public HDFC(String accountNumber, double creditScore, String holderName) {**  **super();**  **this.accountNumber = accountNumber;**  **this.creditScore = creditScore;**  **this.holderName = holderName;**  **}**  **public static double getCREDIT() {**  **return CREDIT;**  **}**  **public static void setCREDIT(double cREDIT) {**  **CREDIT = cREDIT;**  **}**  **public String getAccountNumber() {**  **return accountNumber;**  **}**  **public void setAccountNumber(String accountNumber) {**  **this.accountNumber = accountNumber;**  **}**  **public double getCreditScore() {**  **return creditScore;**  **}**  **public void setCreditScore(double creditScore) {**  **this.creditScore = creditScore;**  **}**  **public String getHolderName() {**  **return holderName;**  **}**  **public void setHolderName(String holderName) {**  **this.holderName = holderName;**  **}**  **private static DecimalFormat df = new DecimalFormat("0.00");**  **private static DecimalFormat dg = new DecimalFormat("0.0");**  **public double calculateCreditScore(double amount) {**  **CREDIT=amount/10;**  **creditScore=creditScore+CREDIT;**  **return creditScore;**    **}**    **public void display() {**  **System.out.println("Amount Paid Successfully !!!");**  **System.out.println("Hi,"+getHolderName());**  **System.out.println("You have gained "+df.format(CREDIT)+" credit score for the payment of "+dg.format(CREDIT\*10));**  **System.out.println("Your Total Credit Score is "+df.format(getCreditScore()));**  **}**  **}** | **import java.text.DecimalFormat;**  **public class ICICI extends RBI {**  **private String accountNumber;**  **private double creditScore;**  **private String holderName;**  **static double CREDIT;**  **public ICICI(String accountNumber, double creditScore, String holderName) {**  **super();**  **this.accountNumber = accountNumber;**  **this.creditScore = creditScore;**  **this.holderName = holderName;**  **}**  **public static double getCREDIT() {**  **return CREDIT;**  **}**  **public static void setCREDIT(double cREDIT) {**  **CREDIT = cREDIT;**  **}**  **public String getAccountNumber() {**  **return accountNumber;**  **}**  **public void setAccountNumber(String accountNumber) {**  **this.accountNumber = accountNumber;**  **}**  **public double getCreditScore() {**  **return creditScore;**  **}**  **public void setCreditScore(double creditScore) {**  **this.creditScore = creditScore;**  **}**  **public String getHolderName() {**  **return holderName;**  **}**  **public void setHolderName(String holderName) {**  **this.holderName = holderName;**  **}**  **private static DecimalFormat df = new DecimalFormat("0.00");**  **private static DecimalFormat dg = new DecimalFormat("0.0");**  **public double calculateCreditScore(double amount) {**  **CREDIT=amount/10;**  **creditScore=creditScore+CREDIT;**  **return creditScore;**    **}**    **public void display() {**  **System.out.println("Amount Paid Successfully !!!");**  **System.out.println("Hi,"+getHolderName());**  **System.out.println("You have gained "+df.format(CREDIT)+" credit score for the payment of "+dg.format(CREDIT\*10));**  **System.out.println("Your Total Credit Score is "+df.format(getCreditScore()));**  **}**  **}** | **public class RBI implements Bank{**  **private String accountNumber;**  **private double creditScore;**  **private String holderName;**  **static double CREDIT;**    **public double calculateCreditScore() {**  **return creditScore;**  **}**  **public void display() {**  **//fill your code here**  **}**  **}** | **import java.io.BufferedReader;**  **import java.io.IOException;**  **import java.io.InputStreamReader;**  **import java.util.Scanner;**  **public class Main {**  **public static String accountNumber;**  **public static double creditScore;**  **public static String holderName;**  **public static void main(String[] args) throws IOException {**  **System.out.println("Select the Bank Name\r\n" + "1.ICICI\r\n" + "2.HDFC");**  **BufferedReader br = new BufferedReader(new InputStreamReader(System.in));**  **int i = Integer.parseInt(br.readLine());**  **switch (i) {**  **case 1:**  **System.out.println("Enter the Holder Name");**  **holderName = br.readLine();**  **System.out.println("Enter the Account Number");**  **accountNumber = br.readLine();**  **System.out.println("Enter the Previous Credit Score");**  **creditScore = Double.parseDouble(br.readLine());**  **ICICI icici = new ICICI(accountNumber, creditScore, holderName);**  **System.out.println("Enter the Amount to be Paid");**  **icici.calculateCreditScore(Double.parseDouble(br.readLine()));**  **icici.display();**  **break;**  **case 2:**  **System.out.println("Enter the Holder Name");**  **holderName = br.readLine();**  **System.out.println("Enter the Account Number");**  **accountNumber = br.readLine();**  **System.out.println("Enter the Previous Credit Score");**  **creditScore = Double.parseDouble(br.readLine());**  **HDFC hdfc = new HDFC(accountNumber, creditScore, holderName);**  **System.out.println("Enter the Amount to be Paid");**  **hdfc.calculateCreditScore(Double.parseDouble(br.readLine()));**  **hdfc.display();**  **break;**  **default:**  **System.out.println("Invalid Bank type");**  **break;**  **}**  **}**  **}** |

**Single Inheritance**

Single inheritance enables a derived class to inherit properties and behavior from a single parent class. It allows a derived class to inherit the properties and behavior of a base class, thus enabling code reusability as well as adding new features to the existing code.  
  
**Strictly adhere to the Object Oriented Specifications given in the problem statement. All class names, member variable names and function names should be the same as specified in the problem statement.**  
  
Create **Event**class with following **protected**member variables

|  |  |
| --- | --- |
| **Data Type** | **Member Variable** |
| string | name |
| string | type |
| string | organizer |

Create **default constructor**and a **parameterized constructor**with arguments in order  **Event(string name, string type, string organizer)**.

Include appropriate **getters**and **setters**.

Create **StageEvent**class which extends **Event**class with following **private**member variables

|  |  |
| --- | --- |
| **Data Type** | **Member Variable** |
| string | show |
| string | startDate |
| string | endDate |

Create **default constructor**and a **parameterized constructor** by invoking the base class constructor

Include appropriate **getters**and **setters**.

Include following method in **StageEvent** class

|  |  |
| --- | --- |
| **Method Name** | **Method Description** |
| void display() | This function is used to display the name,type,organizer,show,startDate and endDate. |

Create a driver class named **Main** to get Event details from user and access the base class using derived class object and call the display method to display the details.

**Input and Output format:**

Refer sample Input and Output for formatting specifications.

**Sample Input and Output :**

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

Enter Event name

**Magic-Show**

Enter Event type

**Entertainment**

Enter organizer

**Kalai**

Enter show details

**ABC\_Magic\_Show**

Enter start time

**11/6/2018**

Enter end time

**15/6/2018**

Event details:

Event name: Magic-show

Event type: Entertainment

Event organizer: Kalai

Show details: ABC\_magic\_show

Start date: 11/6/2018

End date: 15/6/2018

|  |  |  |
| --- | --- | --- |
| **public class Event {**  **protected**  **String name;**  **String type;**  **String organizer;**  **public Event() {**  **super();**  **// TODO Auto-generated constructor stub**  **}**  **public Event(String name, String type, String organizer) {**  **super();**  **this.name = name;**  **this.type = type;**  **this.organizer = organizer;**  **}**  **public String getName() {**  **return name;**  **}**  **public void setName(String name) {**  **this.name = name;**  **}**  **public String getType() {**  **return type;**  **}**  **public void setType(String type) {**  **this.type = type;**  **}**  **public String getOrganizer() {**  **return organizer;**  **}**  **public void setOrganizer(String organizer) {**  **this.organizer = organizer;**  **}**      **}** | **public class StageEvent extends Event {**  **private String show;**  **private String startDate;**  **private String endDate;**  **StageEvent() {**  **super();**  **}**  **public StageEvent(String name, String type, String organizer, String show, String startDate, String endDate) {**  **super(name, type, organizer);**  **this.show = show;**  **this.startDate = startDate;**  **this.endDate = endDate;**  **}**  **public String getShow() {**  **return show;**  **}**  **public void setShow(String show) {**  **this.show = show;**  **}**  **public String getStartDate() {**  **return startDate;**  **}**  **public void setStartDate(String startDate) {**  **this.startDate = startDate;**  **}**  **public String getEndDate() {**  **return endDate;**  **}**  **public void setEndDate(String endDate) {**  **this.endDate = endDate;**  **}**  **void display() {**    **System.out.println("Event details:");**  **System.out.println(**  **"Event name: " + getName() + "\nEvent type: " + getType() + "\nEvent organizer: " + getOrganizer() + "\nShow details: " + getShow() + "\nStart date: " + getStartDate() + "\nEnd date: " + getEndDate());**  **}**  **}** | **import java.util.Scanner;**  **public class Main {**  **public static void main(String[] args) {**  **Scanner sc = new Scanner(System.in);**  **System.out.println("Enter Event name");**  **String ename = sc.nextLine();**  **System.out.println("Enter Event type");**  **String etype = sc.nextLine();**  **System.out.println("Enter organizer");**  **String organizer = sc.next();**  **System.out.println("Enter show details");**  **sc.nextLine();**  **String show = sc.nextLine();**  **System.out.println("Enter start date");**  **String start = sc.next();**  **System.out.println("Enter end date");**  **String end = sc.next();**  **System.out.println("");**    **StageEvent stageEvent = new StageEvent(ename, etype, organizer, show, start, end);**  **stageEvent.display();**  **}**  **}** |

**Abstract Class**

In bank they wanted to issue the authority of the customer .They wanted to state that in what type of credit card the customer is authorized.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, member variable names and function names should be the same as specified in the problem statement.**  
  
Create a **abstract** class **CreditCard** with the following private data member varibales

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| String | cardNumber |
| String | cardType |
| String | holderName |

Include appropriate **getters**and **setters**for the above class  
  
Include the abstract method **display()**in the class **CreditCard**  
   
Create the class **Mastercard** which extends the class **CreditCard** .  
Implement the abstract method in the class **Mastercard.**

|  |  |
| --- | --- |
| **Method name** | **Description** |
| void display() | This method should display the details of the Mastercard. |

Create the class **Visacard** which extends the class **CreditCard** .

Implement the abstract method in the class **Visacard.**

|  |  |
| --- | --- |
| **Method name** | **Description** |
| void display() | This method should display the details of the Visacard. |

Create a driver class named Main. In the main method, create instances for the above mentioned classes and access the display() method.  
  
**[All text in bold corresponds to input and the rest corresponds to output.]  
  
Sample Input and Output 1:**  
Enter the Credit Card Type  
1.Mastercard  
2.Visacard  
**1**  
Enter the Holder Name  
**Vinay Kumar**  
Enter the Card Number  
**56894123**  
Holder Name : Vinay Kumar  
Card Number : 56894123  
You have an Authority of : Mastercard  
  
**Sample Input and Output 2:**  
 Enter the Credit Card Type  
1.Mastercard  
2.Visacard  
**2**  
 Enter the Holder Name  
**Arun Kumar**  
Enter the Card Number  
**676546**  
 Holder Name : Arun Kumar  
Card Number : 676546  
You have an Authority of : Visacard  
  
**Sample Input and Output 3:**  
Enter the Credit Card Type  
1.Mastercard  
2.Visacard  
**5**  
Invalid Account type

|  |  |  |  |
| --- | --- | --- | --- |
| **public abstract class CreditCard {**  **protected**  **String cardNumber;**  **String cardType;**  **String holderName;**    **public String getCardNumber() {**  **return cardNumber;**  **}**  **public void setCardNumber(String cardNumber) {**  **this.cardNumber = cardNumber;**  **}**  **public String getCardType() {**  **return cardType;**  **}**  **public void setCardType(String cardType) {**  **this.cardType = cardType;**  **}**  **public String getHolderName() {**  **return holderName;**  **}**  **public void setHolderName(String holderName) {**  **this.holderName = holderName;**  **}**  **public abstract void display();**    **}** | **public class Visacard extends CreditCard{**  **@Override**  **public void display() {**  **// TODO Auto-generated method stub**  **System.out.println("Holder Name : "+holderName);**  **System.out.println("Card Number : "+cardNumber);**  **System.out.println("You have an Authority of : Visacard");**    **}**  **//fill your code here**  **}** | **public class Mastercard extends CreditCard{**  **@Override**  **public void display() {**  **// TODO Auto-generated method stub**  **System.out.println("Holder Name : "+holderName);**  **System.out.println("Card Number : "+cardNumber);**  **System.out.println("You have an Authority of : Mastercard");**    **}**  **//fill your code here**  **}** | **import java.util.Scanner;**  **public class Main {**  **public static void main(String[] args) throws Exception {**  **//fill your code here**  **Scanner sc=new Scanner(System.in);**  **Mastercard mc=new Mastercard();**  **Visacard vc=new Visacard();**  **int choice;**  **System.out.println("Enter the Credit Card Type\r\n" +**  **"1.Mastercard\r\n" +**  **"2.Visacard");**  **choice=sc.nextInt();**    **if(choice>0 && choice<3) {**  **sc.nextLine();**  **}**  **if(choice==1) {**  **System.out.println("Enter the Holder Name");**  **mc.setHolderName(sc.nextLine());**  **System.out.println("Enter the Card Number");**  **mc.setCardNumber(sc.next());**  **mc.display();**  **}else if(choice==2) {**  **System.out.println("Enter the Holder Name");**  **vc.setHolderName(sc.nextLine());**  **System.out.println("Enter the Card Number");**  **vc.setCardNumber(sc.next());**  **vc.display();**  **}else {**  **System.out.println("Invalid Account type");**  **}**  **}**  **}** |

**Single Inheritance**

Write a program to display details of the Employee using single Inheritance.  
  
Create a class named **Employee** with the following protected data members.

|  |  |
| --- | --- |
| **DataType** | **Data Member** |
| string | name |
| int | experience |
| int | empId |

Include a **default**and **parameterized constructor** in the following order name, experience, empId.

Create a class named **Developer** which inherits **Employee**class with the following public data members.

|  |  |
| --- | --- |
| **DataType** | **Data Member** |
| string | skills |
| string | role |

Include a **default**and **parameterized constructor**in the following order name, experience, empId, skills, role.

Include the below method in the **Developer** class

|  |  |
| --- | --- |
| **Member Function** | **Description** |
| void displayDetails() | This method is used to display all the details of the developer |

Invoke the base constructor as given below

Create a M**ain**class. In the main method, read the inputs and access the displayDetails method.

**Note:**  
Strictly adhere to the object oriented specifications given as part of the problem statement.  
Use the same class names and member variable names.  
Read all the inputs and create instances for the classes in the main.  
  
**Input and Output Format:**  
Refer sample input and output for formatting specifications.  
All text in bold corresponds to input and the rest corresponds to output.  
  
**Sample Input and Output**  
Enter the name  
**John**  
Enter the experience  
**5**  
Enter the employee id  
**21**  
Enter the technical skills of the developer  
**Java,SpringBoot,Mysql**  
Enter the role of the developer  
**Backend**  
Employee Details  
Name : John  
Experience : 5 years  
Employee Id : 21  
Technical Skills : Java,SpringBoot,Mysql  
Role : Backend

|  |  |  |
| --- | --- | --- |
| **public class Employee {**  **protected String name;**  **protected int experience;**  **protected int empId;**    **public Employee() {**  **super();**  **// TODO Auto-generated constructor stub**  **}**  **public Employee(String name, int experience, int empId) {**  **super();**  **this.name = name;**  **this.experience = experience;**  **this.empId = empId;**  **}**    **}** | **public class Developer extends Employee{**  **String skills;**  **String role;**      **public Developer() {**  **super();**  **}**    **public Developer(String name, int experience, int empId, String skills, String role) {**  **super(name, experience, empId);**  **this.skills = skills;**  **this.role = role;**  **}**  **public void displayDetails() {**  **System.out.println("Employee Details");**  **System.out.println("Name : "+name);**  **System.out.println("Experience : "+experience+" years");**  **System.out.println("Employee Id : "+empId);**  **System.out.println("Technical Skills : "+skills);**  **System.out.println("Role : "+role);**  **}**    **}** | **import java.util.Scanner;**  **public class Main {**  **public static void main(String[] args)**  **{**  **Scanner sc = new Scanner(System.in);**    **System.out.println("Enter the name");**  **String name = sc.next();**    **System.out.println("Enter the experience");**  **int experience = sc.nextInt();**    **System.out.println("Enter the employee id");**  **int empId = sc.nextInt();**    **System.out.println("Enter the technical skills of the developer");**  **String skills = sc.next();**    **System.out.println("Enter the role of the developer");**  **String role = sc.next();**    **Developer e = new Developer(name, experience, empId, skills, role);**  **e.displayDetails();**    **}**  **}** |

**Method Overloading**

The simplest example of method overloading is illustrated:  
You would wish to calculate the perimeter of a triangle. Based on the type of triangle, the no of sides would change and hence the parameters required would change. For an Equilateral triangle - only one side is required. For an Isosceles triangle - two side values are required and for a scalene triangle, All three values are required.

The method that needs to be overloaded is calculatePerimeter in Triangle Class. Go Ahead and implement them.

**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, member variable names and function names should be the same as specified in the problem statement.**  
  
Create a class **Triangle**with the following private data members

|  |  |
| --- | --- |
| **Variable name** | **Data Type** |
| sideOne | int |
| sideTwo | int |
| sideThree | int |

Include appropriate **getters**and **setters**.  
  
 Overload the following methods in **Triangle**class,

|  |  |
| --- | --- |
| **Method name** | **Description** |
| int calculatePerimeter(int sideOne) | This function is used to  calculate the perimeter of the equilateral triangle |
| int calculatePerimeter(int sideOne, int sideTwo) | This function is used to  calculate the perimeter of the isosceles triangle |
| int calculatePerimeter(int sideOne, int sideTwo, int sideThree) | This function is used to  calculate the perimeter of the scalene triangle |

Create a driver class **Main**. In the main method, read inputs from the console and create an instance to access the calculatePerimeter function.

**Sample input and output 1:**

1.Equilateral triangle  
2.Isosceles triangle  
3.Normal Triangle  
**1**  
Enter the length of the side  
**90**  
Perimeter is 270

**Sample input and output 2:**

1.Equilateral triangle  
2.Isosceles triangle  
3.Normal Triangle  
**2**  
Enter the length of the leg side  
**30**  
Enter the length of the base side  
**60**  
Perimeter is 120

**Sample input and output 3:**

1.Equilateral triangle  
2.Isosceles triangle  
3.Normal Triangle  
**3**  
Enter the length of the first side  
**20**  
Enter the length of the second side  
**30**  
Enter the length of the third side  
**40**  
Perimeter is 90

|  |  |
| --- | --- |
| **import java.lang.\*;**  **public class Triangle {**  **private int sideOne;**  **private int sideTwo;**  **private int sideThree;**    **public Triangle() {**  **super();**  **// TODO Auto-generated constructor stub**  **}**  **public int getSideOne() {**  **return sideOne;**  **}**  **public void setSideOne(int sideOne) {**  **this.sideOne = sideOne;**  **}**  **public int getSideTwo() {**  **return sideTwo;**  **}**  **public void setSideTwo(int sideTwo) {**  **this.sideTwo = sideTwo;**  **}**  **public int getSideThree() {**  **return sideThree;**  **}**  **public void setSideThree(int sideThree) {**  **this.sideThree = sideThree;**  **}**  **public int computePerimeter(int sideOne)**  **{**  **int equilateralArea = 0;**    **equilateralArea = 3 \* sideOne;**    **return equilateralArea;**    **}**  **public int computePerimeter(int sideOne,int sideTwo)**  **{**  **int isoscelesArea = 0;**    **isoscelesArea = 2 \* sideOne + sideTwo;**    **return isoscelesArea;**    **}**  **public int computePerimeter(int sideOne,int sideTwo,int sideThree)**  **{**  **int scaleneArea = 0;**    **scaleneArea = sideOne + sideTwo + sideThree;**    **return scaleneArea;**    **}**    **}** | **import java.util.\*;**  **public class Main {**  **public static void main(String[] args) throws Exception {**    **Scanner sc = new Scanner(System.in);**    **System.out.println("1.Equilateral triangle\n2.Isosceles triangle\n3.Normal Triangle");**  **int choice = sc.nextInt();**  **Triangle triangle = new Triangle();**  **int perimeter;**    **switch(choice){**    **case 1 : System.out.println("Enter the length of the side");**  **int length = sc.nextInt();**  **perimeter = triangle.computePerimeter(length);**  **System.out.println("Perimeter is " + perimeter);**    **break;**    **case 2 : System.out.println("Enter the length of the leg side");**  **int legLength = sc.nextInt();**  **System.out.println("Enter the length of the base side");**  **int baseLength = sc.nextInt();**  **perimeter = triangle.computePerimeter(legLength,baseLength);**  **System.out.println("Perimeter is " + perimeter);**    **break;**    **case 3 : System.out.println("Enter the length of the first side");**  **int firstLength = sc.nextInt();**  **System.out.println("Enter the length of the second side");**  **int secondLength = sc.nextInt();**  **System.out.println("Enter the length of the third side");**  **int thirdLength = sc.nextInt();**  **perimeter = triangle.computePerimeter(firstLength,secondLength,thirdLength);**  **System.out.println("Perimeter is " + perimeter);**    **break;**    **default: System.out.println("Invalid input");**  **break;**      **}**    **}**    **}** |

**Count number of vowels in the file**

Write a program to read the input from file "sample.txt" using FileReader and display the number of vowels in the given file.  
  
**Rule:**  
The file name should be sample.txt.  
  
**Input format:**  
Give the input as a file which contains characters.  
  
**Output format:**  
The output will be the integer which is number of vowels in the file. Display the output in the console.  
  
**Sample Input file (sample.txt):**  
Welcome to Java Programming  
  
**Sample Output 1:**  
The number of vowels present in file is: 9

|  |
| --- |
| **import java.util.\*;**  **import java.io.File;**  **import java.io.FileNotFoundException;**  **public class Main {**  **public static void main (String[] args) throws FileNotFoundException**  **{**  **File obj = new File("sample.txt");**  **Scanner s = new Scanner(obj);**  **int count = 0;**  **String input = s.nextLine();**  **input = input.toLowerCase();**  **for(int i = 0; i<input.length();i++){**  **char ch = input.charAt(i);**  **if(ch =='a' || ch =='e' || ch =='i' || ch =='o' || ch =='u'){**  **count++;**  **}**  **}**  **System.out.println("The number of vowels present in file is: " +count);**  **}**  **}** |

**ArrayIndexOutOfBoundsException**

**The next prominent exception which you will see is ArrayIndexOutOfBoundsException. It occurs when the program tries to access the array beyond its size. As we know arrays have fixed size. So when you try to use array beyond its size it throws this exception. Let's try to handle this exception.**

**Handling this exception will also prove to be good for our application. For example, if there are only 100(0 – 99) seats in the event and the user tries to book the 105th seat, it will throw the exception message as “Sorry the seat is not available”. So you must handle it to do a specific job.**

**Create an array of size 100 and assume it as seat array. Get the tickets to be booked from the user and handle any exception that occurs in Main Class. At last display all the tickets booked.**

**Input and Output format:**

**The first line of input consists of an integer which corresponds to the number of seats to be booked.**

**The next n lines of input consist of the integer which corresponds to the seat number.**

**Refer to sample Input and Output for formatting specifications.**

**Note: All Texts in bold corresponds to the input and rest are output.**

**Sample Input and Output 1:**

**Enter the number of seats to be booked:**

**5**

**Enter the seat number 1**

**23**

**Enter the seat number 2**

**42**

**Enter the seat number 3**

**65**

**Enter the seat number 4**

**81**

**Enter the seat number 5**

**99**

**The seats booked are:**

**23**

**42**

**65**

**81**

**99**

**Sample Input and Output 2:**

**Enter the number of seats to be booked:**

**4**

**Enter the seat number 1**

**12**

**Enter the seat number 2**

**101**

**Sorry the seat is not available**

|  |
| --- |
| **import java.util.\*;**  **public class Main {**  **public static void main(String args[]) {**    **int seatNo;**  **Scanner sc=new Scanner(System.in);**  **int[] seats=new int[100];**  **System.out.println("Enter the number of seats to be booked:");**  **int noOfSeats=sc.nextInt();**  **int[] bookedArray=new int[noOfSeats];**  **try**  **{**  **//int i=1;**    **for(int index=0;index<noOfSeats;index++)**  **{**  **System.out.println("Enter the seat number "+(index+1));**  **seatNo=sc.nextInt();**  **seats[seatNo-1]=seatNo;**  **bookedArray[index]=seatNo;**  **}**  **System.out.println("The seats booked are:");**  **for(int index=0;index<noOfSeats;index++)**  **{**  **System.out.println(bookedArray[index]);**  **}**  **}**  **catch(ArrayIndexOutOfBoundsException e)**  **{**  **System.out.println("Sorry the seat is not available");**  **}**  **}**  **}** |

**Employee LOP management System**

**Employee LOP management System**

Write a java program to calculate total LOP and display the LOP details of all the employees using abstract class and polymorphism. LOP may be HourlyLop or DailyLop.  
  
**Problem Specifications:**  
**[Note : Strictly adhere to the object oriented specifications given as a part of the problem statement. Use the same class names, attribute names and method names.]**  
  
Create a **abstract** class named **LOP**with the following member variables / attributes

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| employeeId | Integer |
| date | String |
| lopPerDay | static final Integer |
| lopPerHour | static final Integer |

Declare **abstract** method **void calculateLOP( )**inside LOP class  
  
Create a class named **HourlyLop**which **extends** LOP class with the following member variables / attributes

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| noOfHours | Integer |

**Override** **calculateLOP( )** method inside **HourlyLop**class to calculate total Lop for the hours and print LOP details of an employee 

Create a class named **DailyLop** which **extends** LOP class with the following member variables / attributes

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| noOfDays | Integer |

**Override** **calculateLop( )** method inside **DailyLop** class to calculate total Lop for the days and print LOP details of an employee

Create a **Main**class  with main method to test the above classes. Write a do while program to add Lop details for an employee and finally print all lop details using **calculateLop( )** method for all LOP objects.

**Note:**

**'H' -**HourlyLop, **'D'** - DailyLop to differenciate LOP. It must be a 4th parameter in lop details input  
     The static values for LOP value is **500** for **DailyLop** (lopPerDay ) and **100** for **HourlyLop**(lopPerHour) for each employee  
     Total LOP was calculated by **( lopPerDay \* noOfDays)**for DailyLop  and **( lopPerHour \* noOfHours)**for HourlyLop,

**Input order –**employeeId,date,noOfDays/noOfHours,type of LOP

**Output order –**type of LOP :  employeeId,date,noOfDays/noOfHours,total LOP

Refer sample input and output formatting specifications.

**[ All text in bold corresponds to input and the rest corresponds to output.]   
  
Sample Input and Output  :**  

Enter employee details:

**1,02/01/2018,10,H**

Do you want to add another?(Yes/No)

**Yes**

Enter employee details:

**2,02/04/2018,5,D**

Do you want to add another?(Yes/No)

**Yes**

Enter employee details:

**3,05/07/2018,2,H**

Do you want to add another?(Yes/No)

**Yes**

Enter employee details:

**4,05/07/2018,2,D**

Do you want to add another?(Yes/No)

**No**

Hourly LOP : 1,02/01/2018,10,1000

Daily LOP : 2,02/04/2018,5,2500

Hourly LOP : 3,05/07/2018,2,200

Daily LOP : 4,05/07/2018,2,1000

|  |  |  |  |
| --- | --- | --- | --- |
| public abstract class LOP {  int employeeId;  String date;    public LOP() {  super();  // TODO Auto-generated constructor stub  }  static final int lopPerDay=500;  static final int lopPerHour=100;    public int getEmployeeId() {  return employeeId;  }  public void setEmployeeId(int employeeId) {  this.employeeId = employeeId;  }  public String getDate() {  return date;  }  public void setDate(String date) {  this.date = date;  }  public static int getLopperday() {  return lopPerDay;  }  public static int getLopperhour() {  return lopPerHour;  }  public LOP(int employeeId, String date) {  this.employeeId = employeeId;  this.date = date;  }  public abstract void calculateLOP();  } | public class DailyLop extends LOP{  int noOfDays;    public DailyLop() {  super();  // TODO Auto-generated constructor stub  }  public DailyLop(int employeeId, String date,int noOfDays) {  super(employeeId, date);  this.noOfDays=noOfDays;  // TODO Auto-generated constructor stub  }  @Override  public void calculateLOP() {  // TODO Auto-generated method stub  int totalLop=lopPerDay\*noOfDays;  System.out.println("Daily LOP : "+employeeId+","+date+","+noOfDays+","+totalLop);  }  } | public class HourlyLop extends LOP {  int noOfHours;    public int getNoOfHours() {  return noOfHours;  }  public void setNoOfHours(int noOfHours) {  this.noOfHours = noOfHours;  }  public HourlyLop(int employeeId, String date,int noOfHours) {  super(employeeId, date);  this.noOfHours=noOfHours;  // TODO Auto-generated constructor stub  }  @Override  public void calculateLOP() {  // TODO Auto-generated method stub  int totalLop=lopPerHour\*noOfHours;  System.out.println("Hourly LOP : "+employeeId+","+date+","+noOfHours+","+totalLop);  }  } | import java.io.\*;  import java.util.ArrayList;  import java.util.List;  public class Main {  public static void main(String[] args)throws IOException {    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));  String isContinue=null;  List<LOP> lopList = new ArrayList<LOP>();  do {  System.out.println("Enter employee details:");  String details=br.readLine();  String[] token=details.split(",");  String type = token[3];  if(type.equalsIgnoreCase("D")){  LOP a = new DailyLop(Integer.parseInt(token[0]),token[1],Integer.parseInt(token[2]));  lopList.add(a);  }  else if(type.equalsIgnoreCase("H")){  LOP a = new HourlyLop(Integer.parseInt(token[0]),token[1],Integer.parseInt(token[2]));  lopList.add(a);  }  System.out.println("Do you want to add another?(Yes/No)");  isContinue=br.readLine();  }while(isContinue.equalsIgnoreCase("yes"));    for(LOP l:lopList)  {  l.calculateLOP();  }  }  } |

**Bank account management system**

Write a Java program to calculate interest and display the account details of all the account holders using abstract class and polymorphism. An Account may be a CheckingAccount or SavingsAccount.  
  
**Problem Specifications:**  
  
**[Note : Strictly adhere to the object oriented specifications given as a part of the problem statement. Use the same class names, attribute names and method names.]**  
  
Create an **abstract** class named **Account**with the following member variables / attributes   

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| accountNumber | String |
| holderName | String |
| amount | Double |
| interestForSavings | static final int |
| interestForChecking | static final int |

Declare an **abstract** method **void calculateInterest( )**inside **Account class**

Create a class named **CheckingAccount** which **extends** Account class.

**Override** **calculateInterest( )** method inside **CheckingAccount** class to calculate the interest amount and print the account details of an account holder.

Create a class named **SavingsAccount** which **extends** Account class.

**Override** **calculateInterest( )** method inside **SavingsAccount**class to calculate the interest amount and print the account details of an account holder.

Create a **Main**class  with main method to test the above classes. Write a do while program to add Account details for an account holder and finally print all account details using **calculateInterest( )** method for all Account objects.

**Note:**

**'C' -**CheckingAccount , **'S'** - SavingsAccount to differentiate accounts. It must be the 4th parameter in account details input  
     The static final values for rate of interest value is **2** for Savings Account (interestForSavings) and **3** for Checking Account (interestForChecking). Take it as percentage value.  
     Interest Amount was calculated by **( amount \* (rate of interest / 100 ))**

**Input order –**accountNumber,holderName,amount,type of account

**Output order –** type of account :  accountNumber,holderName,amount,calculated interest

Refer sample input and output formatting specifications.

**[ All text in bold corresponds to input and the rest corresponds to output.]**  
  
**Sample Input and Output  :**   
  
    Enter account holder details:  
**61133445,Bala,12000.0,S**  
    Do you want to add another?(Yes/No)  
    **Yes**  
    Enter account holder details:  
**62134345,Suresh,12000.0,C**  
    Do you want to add another?(Yes/No)  
    **Yes**  
    Enter account holder details:  
**71133445,Krish,60000.0,S**  
    Do you want to add another?(Yes/No)  
    **Yes**  
    Enter account holder details:  
**7134435,Babu,60000.0,C**  
    Do you want to add another?(Yes/No)  
    **No**  
    Savings Account : 61133445,Bala,12000.0,240.0  
    Checking Account : 62134345,Suresh,12000.0,360.0  
    Savings Account : 71133445,Krish,60000.0,1200.0  
    Checking Account : 7134435,Babu,60000.0,1800.0

### **Problem Requirements:**

#### **Java**

|  |  |  |
| --- | --- | --- |
| Keyword | Min Count | Max Count |
| abstract | 2 | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| public abstract class Account {  String accountNumber,holderName;  double amount;  double interestForSavings=3.0;  double interestForChecking=2.0;  abstract void calculateInterest();  public String getAccountNumber() {  return accountNumber;  }  public void setAccountNumber(String accountNumber) {  this.accountNumber = accountNumber;  }  public String getHolderName() {  return holderName;  }  public void setHolderName(String holderName) {  this.holderName = holderName;  }  public double getAmount() {  return amount;  }  public void setAmount(double amount) {  this.amount = amount;  }  public void setInterestForSavings(double interestForSavings) {  this.interestForSavings = interestForSavings;  }  public double getInterestForChecking() {  return interestForChecking;  }  public void setInterestForChecking(double interestForChecking) {  this.interestForChecking = interestForChecking;  }  } | public class CheckingAccount extends Account{  public CheckingAccount(String string, String string2, double parseDouble) {  this.accountNumber=string;  this.holderName=string2;  this.amount=parseDouble;  }  @Override  void calculateInterest() {  double interest=this.amount\*(this.interestForChecking/100);  System.out.print("Savings Account : ");  System.out.println(getAccountNumber()+","+getHolderName()+","+getAmount()+","+interest);  }  } | public class SavingsAccount extends Account{  public SavingsAccount(String string, String string2, double parseDouble) {  this.accountNumber=string;  this.holderName=string2;  this.amount=parseDouble;  }  @Override  void calculateInterest() {  // TODO Auto-generated method stub  double a=getAmount()\*(this.interestForSavings/100);  //System.out.println(amount+","+ interestForSavings);  System.out.print("Checking Account : ");  System.out.println(getAccountNumber()+","+getHolderName()+","+getAmount()+","+a);  }  } | import java.io.\*;  import java.util.\*;  public class Main {  public static void main(String[] args)throws IOException {    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));  String isContinue=null;  List<Account> accountList = new ArrayList<Account>();  do {  System.out.println("Enter account holder details:");  String details=br.readLine();  String[] token=details.split(",");  String type = token[3];  if(type.equalsIgnoreCase("S")){  Account a = new CheckingAccount(token[0],token[1],Double.parseDouble(token[2]));  accountList.add(a);  }  else if(type.equalsIgnoreCase("C")){  Account a = new SavingsAccount(token[0],token[1],Double.parseDouble(token[2]));  accountList.add(a);  }  System.out.println("Do you want to add another?(Yes/No)");  isContinue=br.readLine();  }while(isContinue.equalsIgnoreCase("yes"));    for(Account a:accountList)  {  a.calculateInterest();  }  }  } |

**Overloading**

Write a Java program to display the details of the TicketBooking by searching through the ticketBooking list in terms of ticketType and showTime using relationship and polymorphism.

Create a Class **TicketBooking**with the following private attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| showName | String |
| noOfTickets | Integer |
| showTime | Date |
| ticketType | TicketType |

Include appropriate **getters**and **setters**.  
Include **default**and **parameterized constructor**for the class.  
Format for the Parameterized Constructor **TicketBooking(String** **showName, Integer noOfTickets, Date showTime, TicketType ticketType)**  
  
Create a Class **TicketType**with the following private attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| type | String |
| costPerTicket | Double |

Include appropriate **getters**and **setters**.  
Include **default**and **parameterized constructor**for the class.  
Format for the Parameterized Constructor **TicketType(String type, Double costPerTicket)**

Create a class **TicketBookingBO**with the following methods,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| List<TicketBooking> searchTicketBooking(List<TicketBooking> ticketBookingList,String ticketType) | This method accepts a list of TicketBookings and ticketType as arguments and returns a list of TicketBookings that matches with the given ticketType. |
| List<TicketBooking> searchTicketBooking(List<TicketBooking> ticketBookingList,Date showTime) | This method accepts a list of TicketBookings and showTime as arguments and returns a list of TicketBookings that matches with the given showTime. |

The TicketBooking and TicketType details should be given as a comma-separated value in the below order,  
showName, noOfTickets, showTime,type, costPerTicket  
  
When the “TicketBooking ” object is printed, it should display the following details  
Print format:

**System.out.format("%-15s %-20s %-15s %-15s %s\n","Show Name","Show Time","No. of Tickets","Ticket Type","Total cost");**

Create a driver class called  **Main** . In the Main method, obtain input from the user in CSV format and split the input using split method. Display the details of the ticket bookings based on the specified conditions in the same order of inputs given.

**Note:**The ticket booking lists are displayed in the main method.  
            If the search detail is not found, display "Searched booking not found"  
            Display two digits after the decimal point for Double Datatype.

**Sample Input and Output 1:**

Enter the number of bookings  
**5**  
Enter the booking details  
**Singing,3,11-11-2020 10:30:00,Platinum,450  
Dance,4,11-11-2020 15:00:00,Gold,370  
Drama,3,11-11-2020 10:30:00,Silver,320  
FashionLeague,5,11-11-2020 15:00:00,Platinum,570  
Mime,4,11-11-2020 12:00:00,Silver,460**  
1.By ticket type  
2.By show time  
Enter your choice  
**1**  
Enter ticket type  
**Platinum**  
Show Name       Show Time            No. of Tickets  Ticket Type     Total cost  
Singing         11-11-2020 10:30:00  3               Platinum        1350.00  
FashionLeague   11-11-2020 15:00:00  5               Platinum        2850.00

**Sample Input and Output 2:**

Enter the number of bookings  
**5**  
Enter the booking details  
**Singing,3,11-11-2020 10:30:00,Platinum,450  
Dance,4,11-11-2020 15:00:00,Gold,370  
Drama,3,11-11-2020 10:30:00,Silver,320  
FashionLeague,5,11-11-2020 15:00:00,Platinum,570  
Mime,4,11-11-2020 12:00:00,Silver,460**  
1.By ticket type  
2.By show time  
Enter your choice  
**2**  
Enter show time  
**11-11-2020 15:00:00**  
Show Name       Show Time            No. of Tickets  Ticket Type     Total cost  
Dance           11-11-2020 15:00:00  4               Gold            1480.00  
FashionLeague   11-11-2020 15:00:00  5               Platinum        2850.00

**Sample Input and Output 3:**

Enter the number of bookings  
**5**  
Enter the booking details  
**Singing,3,11-11-2020 10:30:00,Platinum,450  
Dance,4,11-11-2020 15:00:00,Gold,370  
Drama,3,11-11-2020 10:30:00,Silver,320  
FashionLeague,5,11-11-2020 15:00:00,Platinum,570  
Mime,4,11-11-2020 12:00:00,Silver,460**  
1.By ticket type  
2.By show time  
Enter your choice  
**2**  
Enter show time  
**11-11-2020 14:00:00**  
Searched booking not found

|  |  |  |  |
| --- | --- | --- | --- |
| import java.util.\*;  public class TicketBooking {  private String showName;  private Integer noOfTickets;  private Date showTime;  private TicketType ticketType;  public TicketBooking() {  super();  // TODO Auto-generated constructor stub  }  public TicketBooking(String showName, Integer noOfTickets, Date showTime, TicketType ticketType) {  super();  this.showName = showName;  this.noOfTickets = noOfTickets;  this.showTime = showTime;  this.ticketType = ticketType;  }  public String getShowName() {  return showName;  }  public void setShowName(String showName) {  this.showName = showName;  }  public Integer getNoOfTickets() {  return noOfTickets;  }  public void setNoOfTickets(Integer noOfTickets) {  this.noOfTickets = noOfTickets;  }  public Date getShowTime() {  return showTime;  }  public void setShowTime(Date showTime) {  this.showTime = showTime;  }  public TicketType getTicketType() {  return ticketType;  }  public void setTicketType(TicketType ticketType) {  this.ticketType = ticketType;  }    } | public class TicketType {  private String type;  private Double costPerTicket;  public TicketType() {  super();  // TODO Auto-generated constructor stub  }  public TicketType(String type, Double costPerTicket) {  super();  this.type = type;  this.costPerTicket = costPerTicket;  }  public String getType() {  return type;  }  public void setType(String type) {  this.type = type;  }  public Double getCostPerTicket() {  return costPerTicket;  }  public void setCostPerTicket(Double costPerTicket) {  this.costPerTicket = costPerTicket;  }    } | import java.util.\*;  public class TicketBookingBO {  List<TicketBooking> OT = new ArrayList<TicketBooking>();  public List<TicketBooking> searchTicketBooking(List<TicketBooking> ticketBookingList,String ticketType){  for (TicketBooking ticketBooking : ticketBookingList) {  if (ticketBooking.getTicketType().getType().equals(ticketType)) {  OT.add(ticketBooking);  }  }  return OT;    }    public List<TicketBooking> searchTicketBooking(List<TicketBooking> ticketBookingList,Date showTime){  for(TicketBooking ticketBooking : ticketBookingList){  if(ticketBooking.getShowTime().equals(showTime)) {  OT.add(ticketBooking);  }  }  return OT;  }  } | import java.io.\*;  import java.text.SimpleDateFormat;  import java.util.\*;  public class Main {  public static void main(String[] args) throws Exception {  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  System.out.println("Enter the number of bookings");  int n = Integer.parseInt(br.readLine());  TicketBooking ticketBooking = new TicketBooking();  TicketBookingBO tbo = new TicketBookingBO();  List<TicketBooking> ticketBookingList = new ArrayList<TicketBooking>();  SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");  System.out.println("Enter the booking details");  for(int i=1;i<=n;i++) {  String input[] = br.readLine().split(",");  TicketType tt = new TicketType(input[3],Double.parseDouble(input[4]));  ticketBooking = new TicketBooking(input[0],Integer.parseInt(input[1]),sdf.parse(input[2]),tt);  ticketBookingList.add(ticketBooking);  }    System.out.println("1.By ticket type\n2.By show time\nEnter your choice");  Integer choice = Integer.parseInt(br.readLine());  List<TicketBooking> searchList = new ArrayList<TicketBooking>();  if(choice==1) {  System.out.println("Enter ticket type");  String type = br.readLine();  searchList = tbo.searchTicketBooking(ticketBookingList, type);  }else if(choice==2) {  System.out.println("Enter show time");  Date showTime = sdf.parse(br.readLine());  searchList = tbo.searchTicketBooking(ticketBookingList, showTime);  }else {  System.out.println("Invalid choice");  }    if(searchList.size()>0) {  System.out.format("%-15s %-20s %-15s %-15s %s\n","Show Name","Show Time","No. of Tickets","Ticket Type","Total cost");  for(TicketBooking tb : searchList) {  Double totalCost = tb.getNoOfTickets()\*tb.getTicketType().getCostPerTicket();  System.out.format("%-15s %-20s %-15d %-15s %.2f\n",tb.getShowName(),sdf.format(tb.getShowTime()),tb.getNoOfTickets(),tb.getTicketType().getType(),totalCost);  }  }else {  System.out.println("Searched booking not found");  }  }  } |

**Add Hall**

**[Note : Strictly adhere to the object oriented specifications given as a part of the problem statement. Follow the naming conventions as mentioned. Create separate classes in separate files.]**

Create a class named **Hall**with the following private attributes/variables.

|  |  |
| --- | --- |
| **Data type** | **Variable** |
| String | name |
| String | contactNumber |
| double | costPerDay |
| String | ownerName |

Include appropriate **getters**and **setters**.  
Include **default**and **parameterized constructor**with parameters in the following order,  
**Hall(string name, string contactNumber, double costPerDay, string ownerName)**  
  
Create a class **HallBO** with the following methods.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void CreateHall(string hallDetails,List<Hall> hallList) | In this method, split the string and create a Hall object and add the hall in hall list. |
| void Display(List<Hall> hallList) | In this method, the hall details are displayed using hall list. |

Create a **Main** class with Main method, to test the above class.  
Get the hall details from the user the perform add operation on list.  
  
**Note:**  
Display the hall details in the following format given in sample output.  
Use the below format given in sample output to display books  
**System.out.format("%s %20s %20s %s\n","Hall name","Contact number","Cost","Owner name");**  
  
**Input format:**  
The input to add the hall details is in the CSV format [**Hall name, Contact number, Cost per day, Owner name**].  
  
**Input and Output Format**  
Refer sample input and output for formatting specifications.  
**All text in bold corresponds to the input and the rest corresponds to output.  
  
Sample Input and Output :**  
Enter the Hall details in CSV format  
**Party hall,9876543210,4000.0,Jarviz**  
Hall created successfully  
Do you want to add more(y/n)?:  
**y**  
Enter the Hall details in CSV format  
**Disco hall,9876543201,5000.0,Starc**  
Hall created successfully  
Do you want to add more(y/n)?:  
**n**  
Hall Details  
Hall name       Contact number                 Cost Owner name  
Party hall           9876543210               4000.0 Jarviz  
Disco hall           9876543201               5000.0 Starc

|  |  |  |  |
| --- | --- | --- | --- |
| class Hall  {  private String name;  private String contactNumber;  private double costPerDay;  private String ownerName;  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getContactNumber() {  return contactNumber;  }  public void setContactNumber(String contactNumber) {  this.contactNumber = contactNumber;  }  public double getCostPerDay() {  return costPerDay;  }  public void setCostPerDay(double costPerDay) {  this.costPerDay = costPerDay;  }  public String getOwnerName() {  return ownerName;  }  public void setOwnerName(String ownerName) {  this.ownerName = ownerName;  }  public Hall(String name, String contactNumber, double costPerDay, String ownerName) {  super();  this.name = name;  this.contactNumber = contactNumber;  this.costPerDay = costPerDay;  this.ownerName = ownerName;  }  } | import java.util.List;  class HallBO  {  public void CreateHall(String hallDetails,List<Hall> hallList){  //fill your code here  String info[]=hallDetails.split(",");  hallList.add(new Hall(info[0],info[1],Double.parseDouble(info[2]),info[3]));  }  void Display(List<Hall> hallList){  System.out.println("Hall Details");  System.out.format("%s %20s %20s %s\n","Hall name","Contact number","Cost","Owner name");  for(Hall hll: hallList)  {  System.out.format("%s %20s %20s %s\n",hll.getName(),hll.getContactNumber(),hll.getCostPerDay(),hll.getOwnerName());  }  }  } | import java.io.\*;  import java.util.ArrayList;  import java.util.List;  import java.util.Scanner;  class Main {  public static void main(String[] args) throws Exception{  ArrayList<Hall> al=new ArrayList<Hall>();  HallBO ob = new HallBO();  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  while(true)  {  System.out.println("Enter the Hall details in CSV format");  String input = br.readLine();  ob.CreateHall(input, al);  System.out.println("Hall created successfully");  System.out.println("Do you want to add more(y/n)?:");  String ans=br.readLine();    if(ans.equalsIgnoreCase("n"))  {    ob.Display(al);  break;  }    }  }  } |  |

**LinkedHashSet**

Write a program to add set of strings to LinkedHashSet and display the set.

Refer sample input and output for formatting specifications.  
**All text in bold corresponds to input and the rest corresponds to output.**  
  
**Sample Input and Output 1:**

Enter n :  
**3**  
Enter strings  
**hi  
hello  
welcome**  
Displaying linked hash set elements :  
hi  
hello  
welcome

|  |
| --- |
| import java.util.LinkedHashSet;  import java.util.Scanner;  public class Main {  public static void main(String[] args)  {  Scanner sc=new Scanner(System.in);  LinkedHashSet<String> lhs=new LinkedHashSet<>();  System.out.println("Enter n :");  int n=sc.nextInt();  String dummy=sc.nextLine();  System.out.println("Enter strings");  for(int i=0;i<n;i++)  {  lhs.add(sc.nextLine());  }  System.out.println("Displaying linked hash set elements :");  for(String string : lhs)  {  System.out.println(string); }  sc.close();  }  } |

**Array List Sorting**

Write a program to add strings to an array list and sort the list in ascending order  
  
**Input & Output Format :**  
The size of the Arraylist must be a integer .  
The values of Array list must be Generic String.

**SAMPLE INPUT & OUTPUT :**  
  
Enter the maximum size of arraylist  
10  
Enter the string values  
Bottle  
Pen  
Apple  
Calender  
Cup  
Mouse  
Keyboard  
Handbag  
Mobile  
Helmet  
  
ArrayList before sorting  
Bottle  
Pen  
Apple  
Calender  
Cup  
Mouse  
Keyboard  
Handbag  
Mobile  
Helmet  
  
ArrayList after sorting  
Apple  
Bottle  
Calender  
Cup  
Handbag  
Helmet  
Keyboard  
Mobile  
Mouse  
Pen

|  |
| --- |
| import java.util.\*;  import java.io.\*;  public class Main {  public static void main(String[] args) throws IOException{  BufferedReader br =new BufferedReader(new InputStreamReader(System.in));  // TODO Auto-generated method stub  System.out.println("Enter the maximum size of arraylist");  int n = Integer.parseInt(br.readLine());  List<String> ar = new ArrayList<>();  System.out.println("Enter the string values");  while(n>0) {  ar.add(br.readLine());  n--;  }    System.out.println("ArrayList before sorting");  for(String val : ar) {  System.out.println(val);  }    Collections.sort(ar);    System.out.println("ArrayList after sorting");  for(String ans : ar) {  System.out.println(ans);  }    }  } |

**Voter ArrayList**

During the year 2020, government going to conduct an election. Due to this, voter list is keep on verifying with the age limit is equal to or greater than 18.

Write a program to read ‘n’ number of user details(name and age) and add the names to the voters list if the age of the candidate is equal to or greater than 18. After added to the list, display the candidates in the voters list.

**Input and Output Format:**  
Refer sample input and output for formatting specifications.  
All text in bold corresponds to input and the rest corresponds to output.  
   
**Sample Input Output :**

Enter no. of candidates:  
**3**  
Enter the name:  
**Vanitha**  
Enter the age:  
**29**  
Enter the name:  
**Prasanna**  
Enter the age:  
**17**  
Enter the name:  
**Kathir**  
Enter the age:  
**18**  
Voter list:  
Vanitha  
Kathir

|  |
| --- |
| import java.io.\*;  import java.util.\*;  public class Main {  public static void main(String[] args) throws IOException  {  BufferedReader br =new BufferedReader(new InputStreamReader(System.in));  System.out.println("Enter no. of candidates:");  int n = Integer.parseInt(br.readLine());  List<String> ar = new ArrayList<>();  List<Integer> num = new ArrayList<>();    while(n>0)  {  System.out.println("Enter the name:");  String name =br.readLine();    System.out.println("Enter the age:");  int age = Integer.parseInt(br.readLine());  if(age>=18)  {  ar.add(name);  }  n--;  }  System.out.println("Voter list:");  for(int i=0;i<ar.size();i++)  {  System.out.println(ar.get(i));  }    }  } |

**Alphabet Pattern 1**

Write a program to print the given pattern.

**Input and Output Format:**

Input consists of a single integer that corresponds to the number of rows,n.

The output is the alphabet pattern for the given input,n.

**Sample Input 1:**

5

**Sample Output 1:**

A

AB

ABC

ABCD

ABCDE

**Sample Input 2:**  
7

**Sample Output 2:**  
A  
AB  
ABC  
ABCD  
ABCDE  
ABCDEF  
ABCDEFG

|  |
| --- |
| import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner s= new Scanner(System.in);  int alphabet=65;  //System.out.println("enter");  int num=s.nextInt();  for (int i=0; i<num; i++) {  for(int j=0; j<=i; j++) {  System.out.print((char)(alphabet+j)+"");  }  System.out.println();  }  }  } |

**Hospital Management System**

**[Note:** **Strictly adhere to the object-oriented specifications given as a part of the problem statement. Follow the naming conventions as mentioned. Create separate classes in separate files.]**

Create class **User** with the following private attributes/variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| String | name |
| String | email |
| String | mobileNumber |
| String | address |

Include parameterized constructor with parameters in the following order,  
**public User(String name, String email, String mobileNumber, String address)**

In the **User**class include the following method

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void display() | In this method, displays user details in the following order name, email, mobileNumber and address. |

Create class **Doctor**which inherits **User** class with the following private attributes/variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| String | qualification |
| int | experience |

Include parameterized constructor with parameters in the following order,   
**Doctor(String name, String email, String mobileNumber, String address, String qualification, int experience)**

In the **Doctor**class include the following method

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void display() | Calls the parent class display() and display the doctor details. |

Create class **Patient**which inherits **User** class with the following private attributes/variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| String | bloodGroup |
| double | height |
| double | weight |

Include parameterized constructor with parameters in the following order,   
**Patient(String name, String email, String mobileNumber, String address, String bloodGroup, double height, double weight)**

In the **Patient**class include the following method

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void display() | Calls the parent class display() and display the patient details. |

Createa driver class**Main**class with **main** method, get user detailsin comma separated format and display the user details using the display() method.  
  
**Input and Output Format**  
Refer sample input and output for formatting specifications.  
**All text in bold corresponds to the input and the rest corresponds to output.**  
  
**Sample Input and Output 1:**  
Menu  
1.Doctor  
2.Patient  
**1**  
Enter the doctor details(name,email,mobile number,address,qualification,experience)  
**Albert,albert@gmail.com,7485961253,Canada,MBBS,5**  
Doctor details  
Name: Albert  
Email: albert@gmail.com  
Mobile number: 7485961253  
Address: Canada  
Qualification: MBBS  
Experience: 5  
  
**Sample Input and Output 2:**  
Menu  
1.Doctor  
2.Patient  
**2**  
Enter the patient details(name,email,mobile number,address,bloodGroup,height,weight)  
**Jonas,jonas@gmail.com,9457961253,Canada,O+,165,55**  
Patient details  
Name: Jonas  
Email: jonas@gmail.com  
Mobile number: 9457961253  
Address: Canada  
BloodGroup: O+  
Height: 165.0  
Weight: 55.0  
  
**Sample Input and Output 3:**  
Menu  
1.Doctor  
2.Patient  
**4**  
Invalid input

|  |  |  |  |
| --- | --- | --- | --- |
| public class Patient extends User {  private String bloodGroup;  private double height;  private double weight;  public Patient() {  super();  }  public Patient(String name, String email, String mobileNumber, String address,String bloodGroup, double height, double weight) {  this.setName(name);  this.setEmail(email);  this.setMobileNumber(mobileNumber);  this.setAddress(address);  this.bloodGroup = bloodGroup;  this.height = height;  this.weight = weight;  }  public String getBloodGroup() {  return bloodGroup;  }  public void setBloodGroup(String bloodGroup) {  this.bloodGroup = bloodGroup;  }  public double getHeight() {  return height;  }  public void setHeight(double height) {  this.height = height;  }  public double getWeight() {  return weight;  }  public void setWeight(double weight) {  this.weight = weight;  }  public void display()  {  System.out.println("Patient details\nName: "+getName()+"\nEmail: "+getEmail()+"\nMobile number: "+getMobileNumber()+"\nAddress: "+getAddress()+"\nBloodGroup: "+bloodGroup+"\nHeight: "+height+"\nWeight: "+weight);    }  } | public class User {  private String name;  private String email;  private String mobileNumber;  private String address;        public User() {  super();  }  public User(String name, String email, String mobileNumber, String address) {    this.name = name;  this.email = email;  this.mobileNumber = mobileNumber;  this.address = address;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public String getMobileNumber() {  return mobileNumber;  }  public void setMobileNumber(String mobileNumber) {  this.mobileNumber = mobileNumber;  }  public String getAddress() {  return address;  }  public void setAddress(String address) {  this.address = address;  }  public void display()  {  //fill your code here  }  } | public class Doctor extends User{  private String qualification;  private int experience;      public Doctor() {  super();  }      public Doctor(String name, String email, String mobileNumber, String address, String qualification, int experience) {  this.setName(name);  this.setEmail(email);  this.setMobileNumber(mobileNumber);  this.setAddress(address);  this.qualification = qualification;  this.experience = experience;  }  public String getQualification() {  return qualification;  }  public void setQualification(String qualification) {  this.qualification = qualification;  }  public int getExperience() {  return experience;  }  public void setExperience(int experience) {  this.experience = experience;  }  //fill your code here  public void display()  {  System.out.println("Doctor details\nName: "+getName()+"\nEmail: "+getEmail()+"\nMobile number: "+getMobileNumber()+"\nAddress: "+getAddress()+"\nQualification: "+getQualification()+"\nExperience: "+getExperience());  }  } | import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner s=new Scanner(System.in);    System.out.println("Menu\n1.Doctor\n2.Patient");  int choice=s.nextInt();    if(choice==1) {  System.out.println("Enter the doctor details(name,email,mobile number,address,qualification,experience)");  String detail=s.next();  String [] d=detail.split(",");  User u=new Doctor(d[0],d[1],d[2],d[3],d[4],Integer.parseInt(d[5]));  u.display();  }  else if(choice==2) {  System.out.println("Enter the patient details(name,email,mobile number,address,bloodGroup,height,weight)");  String detail=s.next();  String [] d=detail.split(",");  User u=new Patient(d[0],d[1],d[2],d[3],d[4],Double.parseDouble(d[5]),Double.parseDouble(d[6]));  u.display();  }  else {  System.out.println("Invalid input");  }  }  } |

**Hall management**

**[Note :  
Strictly adhere to the object oriented specifications given as a part of the problem statement.  
Follow the naming conventions as mentioned. Create separate classes in separate files.]**

Create a class named **Hall**with the following private attributes/variables.

|  |  |
| --- | --- |
| **Data type** | **Variable** |
| String | name |
| String | contactNumber |
| double | costPerDay |
| String | ownerName |

Include appropriate **getters**and **setters**.  
Include **default**and **parameterized constructor**with parameters in the following order,  
**Hall(string name, string contactNumber, double costPerDay, string ownerName)**  
  
Create a class **HallBO** with the following methods.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void createHall(string hallDetails,List<Hall> hallList) | In this method, split the string and create a Hall object and add the hall in hall list. |
| public void removeHall(int index, List<Hall> hallList) | In this method, the hall object present in the given index should be removed from hall list. Then display “Hall removed successfully”. |
| void display(List<Hall> hallList) | In this method, the hall details are displayed using hall list. If no hall present in list then display “The list is empty”. |

Create a **Main** class with Main method, to test the above class.  
Get the hall details from the user the perform add and remove operation on list.  
  
**Note:**  
Display the hall details in the following format given in sample output.  
**Input format:**  
The first line of input is an integer which corresponds to the choice of which action to be done.  
The input to add the hall details is in the CSV format [**Hall name, Contact number, Cost per day, Owner name**].  
The input to remove the hall details is an integer that corresponds to the hall index number. The index starts from 0.  
  
**Input and Output Format**  
Refer sample input and output for formatting specifications.  
**All text in bold corresponds to the input and the rest corresponds to output.  
  
Sample Input and Output 1:**  
Menu  
1.Add Hall  
2.Remove Hall  
Enter your choice  
**1**  
Enter the Hall details in CSV format  
**Party hall,9876543210,4000.0,Jarviz**  
Hall created successfully  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Hall  
2.Remove Hall  
Enter your choice  
**1**  
Enter the Hall details in CSV format  
**Disco hall,9876543201,5000.0,Starc**  
Hall created successfully  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Hall  
2.Remove Hall  
Enter your choice  
**2**  
Enter the index of the hall to be removed  
**0**  
Hall removed successfully  
Do you want to continue(y/n)?:  
**n**  
Hall Details  
Name           ContactNumber  CostPerDay     OwnerName  
Disco hall     9876543201     5000.0           Starc  
  
**Sample Input and Output 2:**

Menu  
1.Add Hall  
2.Remove Hall  
Enter your choice  
**3**  
Invalid choice  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Hall  
2.Remove Hall  
Enter your choice  
**1**  
Enter the Hall details in CSV format  
**Cadogan hall,8765932114,2500.0,Shubha**  
Hall created successfully  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Hall  
2.Remove Hall  
Enter your choice  
**1**  
Enter the Hall details in CSV format  
**Royal hall,9865321470,7000.0,Punitha**  
Hall created successfully  
Do you want to continue(y/n)?:  
**n**  
Hall Details  
Name    ContactNumber  CostPerDay   OwnerName  
Cadogan hall  8765932114  2500.0  Shubha  
Royal hall  9865321470  7000.0  Punitha

**Sample Input and Output 3:**

Menu  
1.Add Hall  
2.Remove Hall  
Enter your choice  
**2**  
The list is empty  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Hall  
2.Remove Hall  
Enter your choice  
**3**  
Invalid choice  
Do you want to continue(y/n)?:  
**n**  
The list is empty

|  |  |  |
| --- | --- | --- |
| class Hall  {  private String name;  private String contactNumber;  private double costPerDay;  private String ownerName;        public Hall() {  super();  }  public Hall(String name, String contactNumber, double costPerDay, String ownerName) {  super();  this.name = name;  this.contactNumber = contactNumber;  this.costPerDay = costPerDay;  this.ownerName = ownerName;  }    public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getContactNumber() {  return contactNumber;  }  public void setContactNumber(String contactNumber) {  this.contactNumber = contactNumber;  }  public double getCostPerDay() {  return costPerDay;  }  public void setCostPerDay(double costPerDay) {  this.costPerDay = costPerDay;  }  public String getOwnerName() {  return ownerName;  }  public void setOwnerName(String ownerName) {  this.ownerName = ownerName;  }  } | import java.util.List;  class HallBO  {  public void createHall(String hallDetails,List<Hall> hallList){  String [] arr=hallDetails.split(",");  Hall h=new Hall(arr[0], arr[1], Double.parseDouble(arr[2]), arr[3]);  hallList.add(h);    }    public void removeHall(int index, List<Hall> hallList){  hallList.remove(index);  System.out.println("Hall removed successfully");  }    void display(List<Hall> hallList){    if(hallList.isEmpty()) {  System.out.println("The list is empty");  }  else {  System.out.println("Hall Details");  //System.out.format("%-10s %-10s %-10s %-10s\n","Name","ContactNumber","CostPerDay","OwnerName");    System.out.println("Name"+" "+"ContactNumber"+" "+"CostPerDay"+" "+"OwnerName");  for(Hall temp: hallList) {  //System.out.format("%-10s %-10s %-10s %-10s",temp.getName(),temp.getContactNumber(),temp.getCostPerDay(),temp.getOwnerName());    System.out.println(temp.getName()+" "+temp.getContactNumber()+" "+temp.getCostPerDay()+" "+temp.getOwnerName());  }  }      }  } | import java.util.ArrayList;  import java.util.List;  import java.util.Scanner;  public class Main {  public static void main(String[] args) throws ArrayIndexOutOfBoundsException{  Scanner s= new Scanner(System.in);  HallBO hb=new HallBO();  List<Hall> hallList=new ArrayList<Hall>();  while(true) {  System.out.println("Menu\n1.Add Hall\n2.Remove Hall\nEnter your choice");  int choice =s.nextInt();    if(choice==1) {  System.out.println("Enter the Hall details in CSV format");  s.nextLine();  String details=s.nextLine();  hb.createHall(details, hallList);  System.out.println("Hall created successfully");  System.out.println("Do you want to continue(y/n)?:");  String ans=s.next();  if(ans.equalsIgnoreCase("n")) {  hb.display(hallList);  System.exit(0);  }  }  //  else if(choice==2) {  s.nextLine();  if(hallList.isEmpty()) {  System.out.println("The list is empty");  }  else{  System.out.println("Enter the index of the hall to be removed");  int index=s.nextInt();  hb.removeHall(index, hallList);  }    System.out.println("Do you want to continue(y/n)?:");  String ans=s.next();  if(ans.equalsIgnoreCase("n")) {  hb.display(hallList);  System.exit(0);  }    }  else {  System.out.println("Invalid choice");  System.out.println("Do you want to continue(y/n)?:");  String ans=s.next();  if(ans.equalsIgnoreCase("n")) {  hb.display(hallList);  System.exit(0);  }  }  }    }  } |

**Search Customer**

Write a Java program by implementing ArrayList and search the customer with the given email.  
  
Create a class **Customer**with following private attributes

|  |  |
| --- | --- |
| **Data Type** | **Variable name** |
| String | name |
| String | email |
| String | phoneNO |

Include appropriate **getters**and **setters**.

Create a class **CustomerBO** with the following methods.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public createCustomer(String customerDetails,List<Customer> customerList) | This method isused to add the customer object into the customerList |
| public void display(List<Customer> customerList) | This method is used to display the list of customers passing as parameter in the given format. |
| public Customer searchCustomerFromList(String email,List<Customer> customerList) | This method is used to search through the customer list for the given email and returns the customer object with the given email |

Create a driver class **Main**. In the main method, test the given scenario

**Note:**

Display the hall details in the following format given in sample output.

If the customer is found for the given email, then display the customer details.   
Otherwise display “**No customer found with given e-mail id**”

**Input format:**  
The first line of input is an integer which corresponds to the choice of which action to be done.  
The input to add the customer details is in the CSV format [**Customer Name, Email, PhoneNO**].  
The input to search the customer details is email id in string

**Input and Output Format**  
Refer sample input and output for formatting specifications.  
**All text in bold corresponds to the input and the rest corresponds to output.**

**Sample Input and Output 1:**

Menu  
1.Add Customer  
2.Display Customer  
3.Search Customer  
Enter your choice  
**1**  
Enter the Customer details in CSV format  
**Madhan,madhan@gmail.com,9597074313**  
Customer created successfully  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Customer  
2.Display Customer  
3.Search Customer  
Enter your choice  
**2**  
Customer Details  
Name    Email  PhoneNO  
Madhan madhan@gmail.com 9597074313  
Do you want to continue(y/n)?:  
**n**

**Sample Input and Output 2:**

Menu  
1.Add Customer  
2.Display Customer  
3.Search Customer  
Enter your choice  
**1**  
Enter the Customer details in CSV format  
**Madhan,madhan@gmail.com,9597074313**  
Customer created successfully  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Customer  
2.Display Customer  
3.Search Customer  
Enter your choice  
**1**  
Enter the Customer details in CSV format  
**Praveen,praveen@gmail.com,9584420136**   
Customer created successfully  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Customer  
2.Display Customer  
3.Search Customer  
Enter your choice  
**3**  
Enter e-mail id to search  
**praveen@gmail.com**  
Customer Details  
Name    Email  PhoneNO  
Praveen praveen@gmail.com 9584420136  
Do you want to continue(y/n)?:  
**n**

**Sample Input and Output 3:**

Menu  
1.Add Customer  
2.Display Customer  
3.Search Customer  
Enter your choice  
**5**  
Invalid choice  
Do you want to continue(y/n)?:  
**n**

|  |  |  |
| --- | --- | --- |
| class Customer  {  public Customer(){}  private String name;  private String email;  private String phoneNO;  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public String getPhoneNO() {  return phoneNO;  }  public void setPhoneNO(String phoneNO) {  this.phoneNO = phoneNO;  }  public Customer(String name, String email, String phoneNO) {  super();  this.name = name;  this.email = email;  this.phoneNO = phoneNO;  }  } | import java.util.List;  public class CustomerBO {    public void createCustomer(String customerDetails,List<Customer> customerList) {  // Fill your code here  String[] p = customerDetails.split(",");  Customer customer = new Customer(p[0],p[1],p[2]);    if(customerList.add(customer)) {  System.out.println("Customer created successfully");  }        }  public void display(List<Customer> customerList) {  // Fill your code here  System.out.println("Customer Details");  System.out.println("Name Email PhoneNO");    for(Customer i: customerList) {  System.out.println(i.getName()+" "+i.getEmail()+" "+i.getPhoneNO());  }    }  public Customer searchCustomerFromList(String email,List<Customer> customerList){  Customer obj = null;  for (Customer customer : customerList) {  if(customer.getEmail().contentEquals(email)) {  obj = customer;  break;  }  }  return obj;    }  } | import java.util.\*;  import java.io.\*;  public class Main {  public static void main(String[] args) throws Exception {  // fill your code here  System.out.println("Menu\n1.Add Customer\n2.Display Customer\n3.Search Customer");  System.out.println("Enter your choice");  Scanner sc = new Scanner(System.in);  int choice = sc.nextInt();  Customer customer = null;  CustomerBO cuBo = new CustomerBO();  List<Customer> customerList = new ArrayList<>();  switch (choice) {  case 1: {  System.out.println("Enter the Customer details in CSV format");  String coDetails = sc.next();  cuBo.createCustomer(coDetails, customerList);  break;  }  case 2: {  cuBo.display(customerList);  break;  }  case 3: {  System.out.println("Enter e-mail id to search");  String email = sc.next();  customer = cuBo.searchCustomerFromList(email, customerList);  System.out.println("Customer Details");  System.out.println("Name Email PhoneNO");  System.out.println(customer.getName()+" "+customer.getEmail()+" "+customer.getPhoneNO());  break;  }  default:  System.out.println("Invalid choice");  }  System.out.println("Do you want to continue(y/n)?:");  String val = sc.next();    while (val.contentEquals("y")) {  System.out.println("Menu\n1.Add Customer\n2.Display Customer\n3.Search Customer");  System.out.println("Enter your choice");  choice = sc.nextInt();  switch (choice) {  case 1: {  System.out.println("Enter the Customer details in CSV format");  String coDetails = sc.next();  cuBo.createCustomer(coDetails, customerList);  break;  }  case 2: {  cuBo.display(customerList);  break;  }  case 3: {  System.out.println("Enter e-mail id to search");  String email = sc.next();  customer = cuBo.searchCustomerFromList(email, customerList);  System.out.println("Customer Details");  System.out.println("Name Email PhoneNO");  System.out.println(customer.getName()+" "+customer.getEmail()+" "+customer.getPhoneNO());  break;  }  default:  System.out.println("Invalid choice");  }    System.out.println("Do you want to continue(y/n)?:");  val = sc.next();  }  }  } |

**Polymorphism with Interface**

Write a Java program to display the final balance amount of an account after a transaction using interface and polymorphism.

**Strictly adhere to the Object Oriented Specifications given in the problem statement. All class names, member variable names and function names should be the same as specified in the problem statement.**

Create an interface named **Banking** and include the following abstract methods in it.

* abstract Payee transferAmount(String name,String bankName,String ifscCode,Double depositAmount)
* abstract Payee transferAmount(String aadharId,String ifscCode,Double depositAmount)
* abstract Payee transferAmount(String UPIid,Double depositAmount)

Create a class **Payee**which implements the interface **Banking**with the following private data member variables.

|  |  |
| --- | --- |
| **Data Type** | **Attributes** |
| String | name |
| String | bankName |
| String | ifscCode |
| String | aadharId |
| String | UPIid |
| Double | balanceAmount |

**Note:**aadharId in India is similar to SSN in USA  
Include appropriate **getters**and **setters**and **constructors**for the above class.

Implement the following overloaded abstract methods in **Payee**class.

|  |  |
| --- | --- |
| **Method name** | **Description** |
| Payee transferAmount(String name,String bankName,String ifscCode,Double depositAmount) | This method accepts payee name, bankName, ifscCode and depositAmount as inputs and checks whether the given details matches the current payee. If it matches, then deposit the amount by adding it to the balanceAmount and return the payee object. Else it returns null. |
| Payee transferAmount(String aadharId,String ifscCode,Double depositAmount) | This method accepts payee aadharId, ifscCode and depositAmount as inputs and checks whether the given details matches the current payee. If it matches, then deposit the amount by adding it to the balanceAmount and return the payee object. Else it returns null. |
| Payee transferAmount(String UPIid,Double depositAmount) | This method accepts payee UPIid and depositAmount as inputs and checks whether the given details matches the current payee. If it matches, then deposit the amount by adding it to the balanceAmount and return the payee object. Else it returns null. |

Create a driver class called  **Main** . In the Main method, read inputs and display the details. If the Payee is matched with the given input then display its details. Otherwise print "**Payee not found**".

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

**Sample Input and Output 1:**

Enter the name  
**Kumar**  
Enter the bankname  
**ICICI**  
Enter the IFSC Code  
**ICIC0000016**  
Enter the aadhar id  
**499118665246**  
Enter the UPI id  
**PAY67882**  
Enter the amount  
**1500**  
1. Transfer using name, bankname and IFSC code  
2. Transfer using aadharId and IFSC code  
3. Transfer using UPI id  
Enter your choice  
**1**  
Enter the payee account name  
**Kumar**  
Enter their bankname  
**ICICI**  
Enter their IFSC Code  
**ICIC0000016**  
Enter the amount to deposit  
**1000**  
Amount deposited successfully  
Current balance of Kumar account is 2500.0

**Sample Input and Output 2:**  
Enter the name  
**Madhu**  
Enter the bankname  
**HDFC**  
Enter the IFSC Code  
**HDFC0000025**  
Enter the aadhar id  
**699118665246**  
Enter the UPI id  
**PAY1234**  
Enter the amount  
**100**  
1. Transfer using name, bankname and IFSC code  
2. Transfer using aadharId and IFSC code  
3. Transfer using UPI id  
Enter your choice  
**2**  
Enter the aadhar id  
**699118665245**  
Enter their IFSC Code  
**HDFC0000025**  
Enter the amount to deposit  
**10**  
Payee not found

**Sample Input and Output 3:**  
Enter the name  
**Madhu**  
Enter the bankname  
**HDFC**  
Enter the IFSC Code  
**HDFC0000025**  
Enter the aadhar id  
**699118665246**  
Enter the UPI id  
**PAY1234**  
Enter the amount  
**100**  
1. Transfer using name, bankname and IFSC code  
2. Transfer using aadharId and IFSC code  
3. Transfer using UPI id  
Enter your choice  
**3**  
Enter the UPI id  
**PAY1234**  
Enter the amount to deposit  
**10**  
Amount deposited successfully  
Current balance of Madhu account is 110.0

|  |
| --- |
| public interface Banking {//fill your code here  abstract Payee transferAmount(String name,String bankName,String ifscCode,Double depositAmount);  abstract Payee transferAmount(String aadharId,String ifscCode,Double depositAmount);  abstract Payee transferAmount(String UPIid,Double depositAmount);  } |
| public class Payee implements Banking{    //fill your code here  private String name;  private String bankName;  private String ifscCode;  private String aadharId;  private String UPIid;  private Double balanceAmount;  //private Double balanceAmount;    public Payee(String name, String bankName, String ifscCode, String aadharId, String uPIid, Double balanceAmount) {  super();  this.name = name;  this.bankName = bankName;  this.ifscCode = ifscCode;  this.aadharId = aadharId;  UPIid = uPIid;  this.balanceAmount = balanceAmount;  }    public Payee() {  super();  // TODO Auto-generated constructor stub  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getBankName() {  return bankName;  }  public void setBankName(String bankName) {  this.bankName = bankName;  }  public String getIfscCode() {  return ifscCode;  }  public void setIfscCode(String ifscCode) {  this.ifscCode = ifscCode;  }  public String getAadharId() {  return aadharId;  }  public void setAadharId(String aadharId) {  this.aadharId = aadharId;  }  public String getUPIid() {  return UPIid;  }  public void setUPIid(String uPIid) {  UPIid = uPIid;  }  public Double getBalanceAmount () {  return balanceAmount;  }  public void setBalance (Double balanceAmount) {  this.balanceAmount = balanceAmount ;  }    public Payee transferAmount(String name,String bankName,String ifscCode,Double depositAmount) {  if(getName().equals(name) && getBankName().equals(bankName)&& getIfscCode().equals(ifscCode)) {  setBalance(depositAmount+getBalanceAmount());  Payee payee = new Payee(getName(),getBankName(),getIfscCode(),getAadharId(),getUPIid(),getBalanceAmount ());  return payee;  }    else {  return null;  }      }  public Payee transferAmount(String aadharId,String ifscCode,Double depositAmount) {  if(getAadharId().equals(aadharId) && getIfscCode().equals(ifscCode)) {  setBalance(depositAmount+getBalanceAmount());  Payee payee = new Payee(getName(),getBankName(),getIfscCode(),getAadharId(),getUPIid(),getBalanceAmount ());  return payee;  }    else {  return null;  }    }  public Payee transferAmount(String UPIid,Double depositAmount) {  if(getUPIid().equals(UPIid)) {  setBalance(depositAmount+getBalanceAmount());  Payee payee = new Payee(getName(),getBankName(),getIfscCode(),getAadharId(),getUPIid(),getBalanceAmount ());  return payee;  }  else {  return null;  }    }    } |
| import java.io.\*;  public class Main {  public static void main(String args[]) throws Exception {  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  System.out.println("Enter the name");  String name = br.readLine();  System.out.println("Enter the bankname");  String bankName = br.readLine();  System.out.println("Enter the IFSC Code");  String ifscCode = br.readLine();  System.out.println("Enter the aadhar id");  String aadharId = br.readLine();  System.out.println("Enter the UPI id");  String upiId = br.readLine();  System.out.println("Enter the amount");  Double balance = Double.parseDouble(br.readLine());  Payee payee = new Payee(name,bankName,ifscCode,aadharId,upiId,balance);  System.out.println("1. Transfer using name, bankname and IFSC code\n2. Transfer using aadharId and IFSC code\n3. Transfer using UPI id\nEnter your choice");  Integer choice = Integer.parseInt(br.readLine());  String pname="",pbank="",pifsc="",paadhar="",pupi="";  Double amount;  Payee p = null;  switch(choice) {  case 1:  System.out.println("Enter the payee account name");  pname = br.readLine();  System.out.println("Enter their bankname");  pbank = br.readLine();  System.out.println("Enter their IFSC Code");  pifsc = br.readLine();  System.out.println("Enter the amount to deposit");  amount = Double.parseDouble(br.readLine());  p = payee.transferAmount(pname, pbank, pifsc, amount);  break;  case 2:  System.out.println("Enter the aadhar id");  paadhar = br.readLine();  System.out.println("Enter their IFSC Code");  pifsc = br.readLine();  System.out.println("Enter the amount to deposit");  amount = Double.parseDouble(br.readLine());  p = payee.transferAmount(paadhar, pifsc, amount);  break;  case 3:  System.out.println("Enter the UPI id");  pupi = br.readLine();  System.out.println("Enter the amount to deposit");  amount = Double.parseDouble(br.readLine());  p = payee.transferAmount(pupi, amount);  break;  default:  System.out.println("Invalid choice");  break;  }  if(p!=null) {  System.out.println("Amount deposited successfully");  System.out.println("Current balance of "+p.getName()+" account is "+p.getBalanceAmount());  }else {  System.out.println("Payee not found");  }    }  } | |

**Alphabet Pattern 4**

Write a program to print the given pattern.

**Input Format:**

Input consists of a single integer which corresponds to the number of rows..

**Output Format:**

Refer sample output.

**Sample Input:**

5

**Sample Output:**

E

ED

EDC

EDCB

EDCBA

|  |
| --- |
| import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner s = new Scanner(System.in);  int input;  //System.out.println("Enter input:");  input = s.nextInt();  int start = (65 + input) - 1;  for (int i = 0; i < input; i++) {  for (int j = 0; j <= i; j++) {  char char\_start = (char) (start - j);  System.out.print(char\_start);  if (j == i) {  System.out.println();  }  }  }  }  } |

**Series-II**

Write a program to generate the below series:

4,32,128,256, ….n

**Input and Output Format:**

The first line is the input consists of a single integer that corresponds to n.

The output consists of the series 4,32,128,…..n separated by a space.

**Sample Input 1:**

4

**Sample Output 1:**

4 32 128 256

**Sample Input 2:**

2

**Sample Output 2:**

4 32

**Sample Input 3:**

6

**Sample Output 3:**

4 32 128 256 256 0

|  |
| --- |
| import java.util.\*;  public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  int n = 4;  int multi = 8;  int input = sc.nextInt();    for(int i=0; i<input; i++)  {  System.out.print(n+" ");  n = n\*multi;  multi = multi/2;  }  }  } |

**More Inhertiance - Mutilevel Inheritance**

Lets explore a scenario in our banking application. Well again, Creditcards, Many types of creditcards are available and most of them have the some set of shared properties and some specific, and many a times due to many types of cards, they are grouped under various levels for a better understanding to pick cards for a purpose. Lets implement a simple multi-level hierarchy as given below.  
  
**CreditCard  
     ---- RewardsCreditCard   
     ---- TravelCreditCard   
                  ---- InternationalTravelCreditCard  
                  ---- CountryTravelCreditCard**  
  
Read the creditcard and travel details from user then calculate the travel amount by using of multilevel inheritance. If the user choose the creditcard type which is not mentioned in list means then display "Invalid Card Type".

Create a class **CreditCard** with following private data members.

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| String | number |
| String | holderName |
| Double | amount |

Use appropriate Getters Setters for **CreditCard** class.

Create the class **TravelCreditcard** which extends the class **CreditCard** with following  private data member

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| Double | exchangePercentage |

Use appropriate Getters Setters for **TravelCreditcard** class.

Create the class **RewardsCreditCard** which extends the class **CreditCard** with following  private data member

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| Double | creditPoints |

Methods in class **RewardsCreditCard**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Function** | **Return Type** |
| calculateAmount(Double amount,Integer numberOfPersons) | Use creditPoints percentage to calculate the payment amount  where user get a discount of 5% of credit points on every ticket. | Double |

Create the class **InternationalCard** which extends the class **TravelCreditcard**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Function** | **Return Type** |
| calculateAmount(Double amount,Integer numberOfPersons) | Use exchange Percentage to calculate the payment amount  where user get a discount of 10% of amount on every ticket. | Double |

Create the class **CountryCard** which extends the class **TravelCreditcard**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Function** | **Return Type** |
| calculateAmount(Double amount,Integer numberOfPersons) | Use exchange Percentage to calculate the payment amount  where user get a discount of 10% of amount on every ticket. | Double |

Use Appropriate Getters Setters for the above classes.

Create a driver class named **Main** which creates an instance of the above mentioned classes.

**Sample Input and Output 1:**

**[All text in bold corresponds to input and the rest corresponds to output.]**  
  
Enter the travel details

Travel Place

**Banglore**

Number of tickets

**2**

Cost per ticket

**1500**

1)Travel Creditcard

2)RewardCreditcard

Enter credit card type

**1**

1)International

2)National

Enter travel creditcard type

**1**

Enter the creditcard number

**123456794**

Enter the creditcard holdername

**Praveen**

Enter the available amount

**65255**

Hello Praveen, You have to pay Rs2700.0

**Sample Input and Output 2:**

Enter the travel details

Travel Place

**Chennai**

Number of tickets

**20**

Cost per ticket

**150**

1)Travel Creditcard

2)RewardCreditcard

Enter credit card type

**3**

Invalid Card Type  
  
**Sample Input and Output 3:**

Enter the travel details  
Travel Place  
**Mumbai**  
Number of tickets  
**3**  
Cost per ticket  
**1000**  
1)Travel Creditcard  
2)RewardCreditcard  
Enter credit card type  
**2**  
Enter the creditcard number  
**465879132**  
Enter the creditcard holdername  
**Jimesh**  
Enter the available amount  
**70000**  
Enter the available rewards  
**60**  
Hello Jimesh, You have to pay Rs2991.0

|  |
| --- |
| public class CreditCard {  private String number;  private String holderName;  private Double amount;      public CreditCard(String number, String holderName, Double amount) {  super();  this.number = number;  this.holderName = holderName;  this.amount = amount;  }  public String getNumber() {  return number;  }  public void setNumber(String number) {  this.number = number;  }  public String getHolderName() {  return holderName;  }  public void setHolderName(String holderName) {  this.holderName = holderName;  }  public Double getAmount() {  return amount;  }  public void setAmount(Double amount) {  this.amount = amount;  }  @Override  public String toString() {  return "CreditCard [number=" + number + ", holderName=" + holderName + ", amount=" + amount + "]";  }  } |
| public class InternationalCard extends TravelCreditcard {  public InternationalCard(String number,String holderName,Double amount,Double exchangePercentage) {  super( number, holderName, amount, exchangePercentage);  }      public InternationalCard(String number, String holderName, Double amount) {  super(number, holderName, amount);  // TODO Auto-generated constructor stub  }    public InternationalCard(String number,String holderName,Double amount, double exchangePercentage) {  super( number, holderName, amount, exchangePercentage);  }  public Double calculateAmount(Double amount,Integer numberOfPersons) {  return amount\* numberOfPersons\*(1-getExchangePercentage());  //fill your code here  }  } |
| public class CountryCard extends TravelCreditcard {    public CountryCard(String number,String holdername,Double amount) {  super(number, holdername, amount);  }    public CountryCard(String number,String holderName,Double amount,Double exchangePercentage) {  super(number,holderName,amount,exchangePercentage);  }  public Double calculateAmount(Double amount,Integer numberOfPersons) {  return amount\*numberOfPersons\*(1-getExchangePercentage());  //fill your code here  }  } |
| public class RewardsCreditCard extends CreditCard {  private Double creditPoints;      public RewardsCreditCard(String number, String holderName,Double amount,Double creditPoints) {  super(number,holderName,amount);  this.creditPoints = creditPoints;  }  public Double getCreditPoints() {  return creditPoints;  }  public void setCreditPoints(Double creditPoints) {  this.creditPoints = creditPoints;  }      @Override  public String toString() {  return "RewardsCreditCard [creditPoints=" + creditPoints + "]";  }  public Double calculateAmount(Double amount,Integer numberOfPersons) {  return (amount\*numberOfPersons)-(creditPoints\*numberOfPersons);  //fill your code here  }  } |
| public class TravelCreditcard extends CreditCard {  private Double exchangePercentage;    public TravelCreditcard(String number, String holderName, Double amount) {  super(number, holderName, amount);  // TODO Auto-generated constructor stub  }    public TravelCreditcard(String number, String holderName, Double amount, Double exchangePercentage) {  super(number, holderName, amount);  this.exchangePercentage = exchangePercentage;  }  public Double getExchangePercentage() {  return exchangePercentage;  }  public void setExchangePercentage(Double exchangePercentage) {  this.exchangePercentage = exchangePercentage;  }  @Override  public String toString() {  return "TravelCreditcard [exchangePercentage=" + exchangePercentage + "]";  }        } |
| import java.util.Scanner;  public class Main {​  public static void main(String[] args) {​  //fill your code here  String number;  String holderName;  Double amount;  Double exchangePercentage = 0.10;  Double creditPoints = 0.05;  int availRewards;  Scanner sc = new Scanner(System.in);  System.out.println("Enter the travel details\n" + "Travel Place");  sc.next();  System.out.println("Number of tickets");  Integer no\_Of\_Tickets =sc.nextInt();  System.out.println("Cost per ticket");  Double cost\_Per\_Ticket =sc.nextDouble();  System.out.println("1)Travel Creditcard\n" + "2)RewardCreditcard\r\n"+ "Enter credit card type");  int card\_Type = sc.nextInt();  switch (card\_Type) {​  case 1:  System.out.println("1)International\n"+ "2)National\n"+ "Enter travel creditcard type");  int travel\_Type = sc.nextInt();  System.out.println("Enter the creditcard number");  number = sc.next();  System.out.println("Enter the creditcard holdername");  holderName = sc.next();  System.out.println("Enter the available amount");  amount = sc.nextDouble();  if(travel\_Type==1) {​  InternationalCard ic = new InternationalCard(number,holderName,amount,exchangePercentage);  System.out.println("Hello "+ holderName+","+ " You have to pay Rs"+ ic.calculateAmount(cost\_Per\_Ticket,no\_Of\_Tickets));  }​  else if(travel\_Type==2) {​  CountryCard cc = new CountryCard(number,holderName,amount,exchangePercentage);  System.out.println("Hello "+ holderName+ ","+" You have to pay Rs"+ cc.calculateAmount(cost\_Per\_Ticket, no\_Of\_Tickets));  }​  else {​  System.out.println("Invalid Card Type");  }​  break;  case 2:  System.out.println("Enter the creditcard number");  number = sc.next();  System.out.println("Enter the creditcard holdername");  holderName = sc.next();  System.out.println("Enter the available amount");  amount = sc.nextDouble();  System.out.println("Enter the available rewards");  availRewards = sc.nextInt();  RewardsCreditCard rc = new RewardsCreditCard (number,holderName,amount,creditPoints\*availRewards);  System.out.println("Hello "+ holderName+ ","+ " You have to pay Rs"+ rc.calculateAmount(cost\_Per\_Ticket, no\_Of\_Tickets));  break;  default:  System.out.println("Invalid Card Type");  break;  }​  }​  } |
|  |

**Inheritance- Airport**

Airports Authority of India (AAI) aims to register the details of all the airports in India. For security reasons, they wanted to have an account of the number of passengers travelling each day and for the cargo , the number of cargo units shipped. Airports have a unique IATA (Internatinal Air Transport Association) Airport Code.

Given the number of flights per day, write a program to calculate the Number of passengers travelled and the number of cargo units shipped per day.

Create a class named **Airport**

Include the following private data members / attributes:

|  |  |
| --- | --- |
| **Data type** | **Variable name** |
| String | name |
| String | city |
| Integer | noOfFlights |

Include appropriate **getters**and **setters**.  
Include **3 argumented constructor** in the order (name,city, noOfFlights)

Create a class named **CommercialServiceAirport** that extends **Airport**

Include the following private data members / attributes:

|  |  |
| --- | --- |
| **Data type** | **Variable name** |
| Integer | noOfpassengersperflight |

Include appropriate **getters**and **setters**.  
Include 4**argumented constructor** in the order (name,city,noOfFlights,noOfpassengersperflight)

Include the following method in the **CommercialServiceAirport** class

|  |  |
| --- | --- |
| **Method name** | **Method description** |
| Integer calculatenoOfPassengers() | This method will calculate and returns the number of passengers travelled. |

Create a class named **CargoServiceAirport** that extends **Airport**

Include the following protected data members / attributes:

|  |  |
| --- | --- |
| **Data type** | **Variable name** |
| Integer | noOfcargoUnitsperflight |

Include appropriate **getters**and **setters**.

Include 4**argumented constructor** in the order (name,city,noOfFlights,noOfcargoUnitsperflight)

Include the following method in the **CargoServiceAirport** class

|  |  |
| --- | --- |
| **Method name** | **Method description** |
| Integer calculatenoOfCargounits() | This method will calculate and returns the number of cargo units shipped. |

Create a driver class **Main** to test the above scenario.

**Note :**

Number of passengers travelled/day=number of flights per day\*number of passengers travelling in each flight

Number of cargo units shipped/day=number of flights per day\*number of cargo units shipped in each flight

**Input and Output Format:**

Refer sample input and output for formatting specifications.

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

**Sample Input and Output:**  
Enter the name

**Kempegowda International Airport**

Enter the city

**Bangalore**

Enter the number of flights per day

**4**

Enter the number of passengers travelling in each flight

**60**

Enter the number of cargo units shipped in each flight

**50**

Airport Details

Number of passengers travelled/day: 240

Number of cargo units shipped/day: 200

|  |
| --- |
| public class Airport{  private String name;  private String city;  private Integer noOfFlights;  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getCity() {  return city;  }  public void setCity(String city) {  this.city = city;  }  public Integer getNoOfFlights() {  return noOfFlights;  }  public void setNoOfFlights(Integer noOfFlights) {  this.noOfFlights = noOfFlights;  }    public Airport() {  super();  // TODO Auto-generated constructor stub  }    public Airport(String name, String city, Integer noOfFlights) {  super();  this.name = name;  this.city = city;  this.noOfFlights = noOfFlights;  }    } |
| public class CommercialServiceAirport extends Airport{  private Integer noOfpassengersperflight;  public Integer getNoOfpassengersperflight() {  return noOfpassengersperflight;  }  public void setNoOfpassengersperflight(Integer noOfpassengersperflight) {  this.noOfpassengersperflight = noOfpassengersperflight;  }  public CommercialServiceAirport(String name, String city, Integer noOfFlights, Integer noOfpassengersperflight) {  super(name, city, noOfFlights);  this.noOfpassengersperflight = noOfpassengersperflight;  }  public Integer calculatenoOfPassengers() {  Integer totalp = getNoOfFlights()\*getNoOfpassengersperflight();  return totalp;  }  } |
| public class CargoServiceAirport extends Airport{  private Integer noOfcargoUnitsperflight;  public Integer getNoOfcargoUnitsperflight() {  return noOfcargoUnitsperflight;  }  public void setNoOfcargoUnitsperflight(Integer noOfcargoUnitsperflight) {  this.noOfcargoUnitsperflight = noOfcargoUnitsperflight;  }  public CargoServiceAirport(String name, String city, Integer noOfFlights, Integer noOfcargoUnitsperflight) {  super(name, city, noOfFlights);  this.noOfcargoUnitsperflight = noOfcargoUnitsperflight;  }  public Integer calculatenoOfCargounits() {  Integer totalc = getNoOfFlights()\*getNoOfcargoUnitsperflight();  return totalc;  }  } |
| import java.util.Scanner;  public class Main {  public static void main(String args[]) throws Exception  {  Scanner sc = new Scanner (System.in);  System.out.println("Enter the name");  String airport = sc.nextLine();  System.out.println("Enter the city");  String city = sc.nextLine();  System.out.print("Enter the number of flights per day");  Integer flight = sc.nextInt();  System.out.println("Enter the number of passengers travelling in each flight");  Integer passengers = sc.nextInt();  CommercialServiceAirport commerciala = new CommercialServiceAirport (airport, city, flight, passengers);    System.out.println("Enter the number of cargo units shipped in each flight");  Integer cargounits = sc.nextInt();  CargoServiceAirport cargoa = new CargoServiceAirport (airport, city, flight, cargounits);  System.out.println("Airport Details");  System.out.println("Number of passengers travelled/day:"+commerciala.calculatenoOfPassengers());  System.out.println("Number of cargo units shipped/day:"+cargoa.calculatenoOfCargounits());  }  } |

**SECRET 7**

The Secret 7 is a secret investigation team consisting of  7 members. The team members meet once every fortnight. The Famous 5 is a rival gang and they  try to steal secrets from secret seven. In each of their meets they decide a passcode for the next meet. The passcode they set is a 4 digit number with same numbers in the even places and odd places. Your friend is a part of the group and seeks your help to identify the Secret 7 members. Can you help him out ???  
  
**Input format:**

Input consists of an integer corresponding to the passcode entered by the member.

**Output format:**

The Output consists of the strings "Passcode matched. Hi Secret 7!!!” or “Sorry!!! passcode mismatched. Wrong identification.”.

Refer sample input and output for formatting specifications.

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

**Sample Input and Output 1:**

Passcode:

**1231**

Sorry!!! passcode mismatched. Wrong identification.

**Sample Input and Output 2:**

Passcode:

**1010**

Passcode matched. Hi Secret 7!!!

|  |
| --- |
| import java.util.Scanner;  public class Main {  public static void main(String[] args) {  Scanner sc = new Scanner(System.in);  System.out.println("Passcode:");  String s = sc.next();  char[] ch = s.toCharArray();    char even = ch[0];  char odd = ch[1];    if(even==ch[2] && odd==ch[3]) {  System.out.println("Passcode matched. Hi Secret 7!!!");  }  else {  System.out.println("Sorry!!! passcode mismatched. Wrong identification.");  }    }    } |

**Customer – SubList**

Write a Java program by implementing ArrayList, add the customers to the list and get the sublist of customers whose name starts with the given character.  
  
Create a class **Customer**with following private attributes

|  |  |
| --- | --- |
| **Data Type** | **Variable name** |
| String | name |
| String | email |
| String | phoneNO |

Include appropriate **getters**and **setters**.

Create a class **CustomerBO** with the following methods.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void createCustomer(String customerDetails,ArrayList<Customer> customerList) | This method is used to split the given customerDetails and add the Customer object to the given customerList. |
| public ArrayList<Customer> searchCustomerFromList(ArrayList<Customer> customerList, String searchCharacter) | This method is used to search through the customer list for the given character and returns the list of customers who are matching the given character |
| public void display(ArrayList<Customer> customerList) | This method is used to display the list of customers passing as parameter in the given format. |

Create a driver class **Main**. In the main method, test the given scenario. Print the statement “**List of Customers:**” in the main method itself.

**Note:**

Use the formatting "**%-20s %-20s %s\n**" while displaying the customer details.

**Input and Output Format:**

Refer sample input and output for formatting specifications.

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

**Sample Input / Output :**  
Menu  
1.Add Customer  
2.Get Search list  
Enter your choice  
**1**  
Enter the Customer details in CSV format  
**Madhan,madhan@gmail.com,9597074313**  
Customer created successfully  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Customer  
2.Get Search list  
Enter your choice  
**4**  
Invalid choice  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Customer  
2.Get Search list  
Enter your choice  
**1**  
Enter the Customer details in CSV format  
**Arun,arun@microsoft.com,8147563201**  
Customer created successfully  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Customer  
2.Get Search list  
Enter your choice  
**1**  
Enter the Customer details in CSV format  
**Jimesh,jimesh@gmail.com,9677523685**  
Customer created successfully  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Customer  
2.Get Search list  
Enter your choice  
**1**  
Enter the Customer details in CSV format  
**Malini,malini@gmail.com,9685523104**  
Customer created successfully  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Customer  
2.Get Search list  
Enter your choice  
**1**  
Enter the Customer details in CSV format  
**Anandh,anandh@yahoo.com,9600761548**  
Customer created successfully  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Customer  
2.Get Search list  
Enter your choice  
**1**  
Enter the Customer details in CSV format  
**Anu,anu@redmail.com,9663257418**  
Customer created successfully  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Customer  
2.Get Search list  
Enter your choice  
**1**  
Enter the Customer details in CSV format  
**Collin,collin@gmail.com,9563112000**  
Customer created successfully  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Customer  
2.Get Search list  
Enter your choice  
**2**  
List of Customers:  
Customer Name        Email                Phone Number  
Madhan               madhan@gmail.com     9597074313  
Arun                 arun@microsoft.com   8147563201  
Jimesh               jimesh@gmail.com     9677523685  
Malini               malini@gmail.com     9685523104  
Anandh               anandh@yahoo.com     9600761548  
Anu                  anu@redmail.com      9663257418  
Collin               collin@gmail.com     9563112000  
Enter the character to find sub list  
**A**  
Customer Name        Email                Phone Number  
Arun                 arun@microsoft.com   8147563201  
Anandh               anandh@yahoo.com     9600761548  
Anu                  anu@redmail.com      9663257418  
Do you want to continue(y/n)?:  
**n**

|  |
| --- |
| public class Customer {  private  String name;  String email;  String phoneNO;      public Customer(String name, String email, String phoneNO) {  super();  this.name = name;  this.email = email;  this.phoneNO = phoneNO;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getEmail() {  return email;  }  public void setEmail(String email) {  this.email = email;  }  public String getPhoneNO() {  return phoneNO;  }  public void setPhoneNO(String phoneNO) {  this.phoneNO = phoneNO;  }      } |
| import java.util.ArrayList;  public class CustomerBO {    public void createCustomer(String customerDetails,ArrayList<Customer> customerList) {  //fill your code here  String[] parts = customerDetails.split(",");  Customer customer = new Customer(parts[0],parts[1],parts[2]);  customerList.add(customer);  }    public ArrayList<Customer> searchCustomerFromList(ArrayList<Customer> customerList, String searchCharacter){  ArrayList<Customer> resultList = new ArrayList<Customer>();  for (Customer customer : customerList) {  if(customer.getName().startsWith(searchCharacter)) {  resultList.add(customer);  }  }  return resultList;  }    public void display(ArrayList<Customer> customerList) {  //fill your code here  System.out.println("List of Customers:");  System.out.format("%-20s %-20s %s\n","Customer Name","Email","Phone Number");  for (Customer customer : customerList) {  System.out.format("%-20s %-20s %s\n",customer.getName(),customer.getEmail(),customer.getPhoneNO());  }  }  } |
| import java.util.\*;  import java.io.\*;  public class Main {  public static void main(String[] args) throws Exception {  // fill your code here  Scanner s = new Scanner(System.in);  String response, customerDetails, searchChar;  ArrayList<Customer> customers = new ArrayList<>();  ArrayList<Customer> searchResults = new ArrayList<>();  CustomerBO customerBO;  int option;  while (true) {  System.out.println("Menu");  System.out.println("1.Add Customer");  System.out.println("2.Get Search list");  System.out.println("Enter your choice");  option = s.nextInt();  switch (option) {  case 1:  System.out.println("Enter the Customer details in CSV format");  customerDetails = s.next();  customerBO = new CustomerBO();  customerBO.createCustomer(customerDetails, customers);  System.out.println("Customer created successfully");  break;  case 2:  customerBO = new CustomerBO();  customerBO.display(customers);  System.out.println("Enter the character to find sub list");  searchChar = s.next();  searchResults = customerBO.searchCustomerFromList(customers, searchChar);  System.out.format("%-20s %-20s %s\n","Customer Name","Email","Phone Number");  for (Customer customer : searchResults) {  System.out.format("%-20s %-20s %s\n",customer.getName(),customer.getEmail(),customer.getPhoneNO());  }  break;  default:  System.out.println("Invalid choice");  break;  }  System.out.println("Do you want to continue(y/n)?:");  response = s.next();  if (response.equalsIgnoreCase("N")) {  break;  }  }  }  } |

**Parking Lot - Requirement**

XpressCity Mall is the recent attraction for the people in the City amidst this pandemic. This mall is a one stop place for all needs including groceries, clothing, beauty supplies, home needs, etc., The mall is been tightly supervised and maintained during these times to ensure people can do their shopping safe and sound.   
  
The parking lot is also part of this vigilance. Since the parking area is widespread, the Mall authorities wanted one unified automatic system to manage the entry and exit of vehicles. The system should collect and store all the details of vehicles entering inside the lot like Registration number, type of vehicle, Name of vehicle, Weight, Parked time, etc., It should then retrieve and display the vehicle details based on its Type or Parked time.   
  
Write a code to fetch the necessary details of the vehicles, search and display the vehicles based on type and parkedTime.  
  
a) Create a Class Vehicle with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| registrationNo | String |
| name | String |
| type | String |
| weight | Double |
| ticket | Ticket |

Mark all the attributes as private, Create / Generate appropriate Getters & Setters, Add a default constructor and a parameterized constructor to take in all attributes in the given order: Vehicle( String registrationNo, String name, String type, Double weight ,Ticket ticket)  
  
b) Create a Class Ticket with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| ticketNo | String |
| parkedTime | Date |
| cost | Double |

Mark all the attributes as private, Create / Generate appropriate Getters & Setters, Add a default constructor and a parameterized constructor to take in all attributes in the given order: Ticket( String ticketNo,  Date parkedTime, Double cost)  
  
c) Create a class **VehicleBO**with the following methods,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public List<Vehicle> findVehicle(List<Vehicle> vehicleList,String type) | This method accepts a list of vehicles and type as arguments and returns a list of vehicles that matches with the given type. |
| public List<Vehicle> findVehicle(List<Vehicle> vehicleList,Date parkedTime) | This method accepts a list of vehicles and parkedTime as arguments and returns a list of vehicles that matches with the given parkedTime. |

The Vehicle and Ticket details should be given as a comma-separated value in the below order,  
registrationNo, name, type, weight,ticketNo,parkedTime,cost  
  
When the “vehicle” object is printed, it should display the following details  
Print format:  
**System.out.format(**"%-15s %-10s %-12s %-7s %s\n","Registration No","Name","Type","Weight","Ticket No"**);**

**Note:**The vehicle lists are displayed in the main method.  
            If any other choice is selected, display "**Invalid Choice**"  
            If the search detail is not found, display "No such vehicle is present"  
            Display one digit after the decimal point for Double Datatype.  
  
**Sample test case 1:**  
Enter the number of vehicles:

6

**AP 19 QS 4556,R3,TwoWheeler,196,A1-002,10-05-2018 11:05:21,100**

**MP 01 LK 0001,Hornet,TwoWheeler,163,A1-009,09-05-2018 08:13:24,50**

**MH 23 F 7856,Gixer,TwoWheeler,221,A1-013,11-05-2018 05:21:40,75**

**GA 45 RF 9515,Duke,TwoWheeler,240,A1-024,10-05-2018 05:16:26,75**

**GJ 83 AX 0545,Ciaz,FourWheeler,530,B4-030,10-05-2018 08:25:33,150**

**HR 46 S 4523,Swift,FourWheeler,846,B4-021,09-05-2018 07:14:13,200**

Enter a search type:

1.By type

2.By parked time

**1**

Enter the vehicle type

**FourWheeler**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Registration No | Name | Type | Weight | Ticket No |
| GJ 83 AX 0545 | Ciaz | FourWheeler | 530.0 | B4-030 |
| HR 46 S 4523 | Swift | FourWheeler | 846.0 | B4-021 |

**Sample test case 2:**

Enter the number of vehicles:

**6**

**AP 19 QS 4556,R3,TwoWheeler,196,A1-002,10-05-2018 11:05:21,100**

**MP 01 LK 0001,Hornet,TwoWheeler,163,A1-009,09-05-2018 08:13:24,50**

**MH 23 F 7856,Gixer,TwoWheeler,221,A1-013,11-05-2018 05:21:40,75**

**GA 45 RF 9515,Duke,TwoWheeler,240,A1-024,11-05-2018 05:21:40,75**

**GJ 83 AX 0545,Ciaz,FourWheeler,530,B4-030,10-05-2018 08:25:33,150**

**HR 46 S 4523,Swift,FourWheeler,846,B4-021,09-05-2018 07:14:13,200**

Enter a search type:

1.By type

2.By parked time

**2**

Enter the parked time:

**11-05-2018 05:21:40**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Registration No | Name | Type | Weight | Ticket No |
| MH 23 F 7856 | Gixer | TwoWheeler | 221.0 | A1-013 |
| GA 45 RF 9515 | Duke | TwoWheeler | 240.0 | A1-024 |

|  |
| --- |
| public class Vehicle {  private String registrationNo;  private String name;  private String type;  private Double weight;  private Ticket ticket;  public Vehicle(String registrationNo, String name, String type,Double weight, Ticket ticket) {  super();  this.registrationNo = registrationNo;  this.name = name;  this.type = type;  this.weight = weight;  this.ticket = ticket;  }  public String getRegistrationNo() {  return registrationNo;  }  public void setRegistrationNo(String registrationNo) {  this.registrationNo = registrationNo;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getType() {  return type;  }  public void setType(String type) {  this.type = type;  }  public Ticket getTicket() {  return ticket;  }  public void setTicket(Ticket ticket) {  this.ticket = ticket;  }  public Double getWeight() {  return weight;  }  public void setWeight(Double weight) {  this.weight = weight;  }    } |
| import java.util.ArrayList;  import java.util.Date;  import java.util.List;  public class VehicleBO {  public List<Vehicle> findVehicle(List<Vehicle> vehicleList, String type) {  ArrayList<Vehicle> result = new ArrayList<Vehicle>();  for (Vehicle vehicle : vehicleList) {  if(vehicle.getType().equals(type)) {  result.add(vehicle);  }  }  return result;  }  public List<Vehicle> findVehicle(List<Vehicle> vehicleList, Date parkedTime) {  ArrayList<Vehicle> result = new ArrayList<Vehicle>();  for (Vehicle vehicle : vehicleList) {  if(vehicle.getTicket().getParkedTime().equals(parkedTime)) {  result.add(vehicle);  }  }  return result;  }  } |
| import java.util.Date;  public class Ticket {  private String ticketNo;  private Date parkedTime;  private Double cost;  public Ticket(String ticketNo, Date parkedTime, Double cost) {  super();  this.ticketNo = ticketNo;  this.parkedTime = parkedTime;  this.cost = cost;  }  public String getTicketNo() {  return ticketNo;  }  public void setTicketNo(String ticketNo) {  this.ticketNo = ticketNo;  }  public Date getParkedTime() {  return parkedTime;  }  public void setParkedTime(Date parkedTime) {  this.parkedTime = parkedTime;  }  public Double getCost() {  return cost;  }  public void setCost(Double cost) {  this.cost = cost;  }  } |
| import java.io.BufferedReader;  import java.io.IOException;  import java.io.InputStreamReader;  import java.text.DateFormat;  import java.text.ParseException;  import java.text.SimpleDateFormat;  import java.util.ArrayList;  import java.util.Date;  import java.util.List;  public class Main {  public static void main(String[] args) throws NumberFormatException, IOException, ParseException {  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  DateFormat df = new SimpleDateFormat("dd-MM-yyyy-HH:mm:ss");  System.out.println("Enter the number of vehicles:");  int count = Integer.parseInt(br.readLine());  ArrayList<Vehicle> vehicles = new ArrayList<Vehicle>();  for (int i = 0; i < count; i++) {  String line = br.readLine();  String[] splitLine = line.split(",");  Date myDate = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss").parse(splitLine[5]);  Ticket tic = new Ticket(splitLine[4], myDate, Double.parseDouble(splitLine[6]));  Vehicle v = new Vehicle(splitLine[0], splitLine[1], splitLine[2], Double.parseDouble(splitLine[3]), tic);  vehicles.add(v);  }  System.out.println("Enter a search type:");  System.out.println("1.By type");  System.out.println("2.By parked time");  int choice = Integer.parseInt(br.readLine());  switch (choice) {  case 1:  System.out.println("Enter the vehicle type:");  String vType = br.readLine();  VehicleBO bo = new VehicleBO();  List<Vehicle> resultOne = bo.findVehicle(vehicles, vType);  if (resultOne.size() > 0) {  System.out.format("%-15s %-10s %-12s %-7s %s\n", "Registration No", "Name", "Type", "Weight",  "Ticket No");  for (Vehicle vehicle : resultOne) {  System.out.format("%-15s %-10s %-12s %-7s %s\n", vehicle.getRegistrationNo(), vehicle.getName(), vehicle.getType(), vehicle.getWeight(),  vehicle.getTicket().getTicketNo());  }  } else {  System.out.println("No such vehicle is present");  }  break;  case 2:  System.out.println("Enter the parked time:");  String vParked = br.readLine();  Date myDateParked = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss").parse(vParked);  VehicleBO bo2 = new VehicleBO();  List<Vehicle> resultTwo = bo2.findVehicle(vehicles, myDateParked);  if (resultTwo.size() > 0) {  System.out.format("%-15s %-10s %-12s %-7s %s\n", "Registration No", "Name", "Type", "Weight",  "Ticket No");  for (Vehicle vehicle : resultTwo) {  System.out.format("%-15s %-10s %-12s %-7s %s\n", vehicle.getRegistrationNo(), vehicle.getName(), vehicle.getType(), vehicle.getWeight(),  vehicle.getTicket().getTicketNo());  }  } else {  System.out.println("No such vehicle is present");  }  break;  default:  System.out.println("Invalid Choice");  break;  }  }  } |