Basic Elements of Java / Practice / Mandatory

**Customized Welcome Message**

Nikhil, the founder of “Pine Tree” company wished to design an Event Management System that would let its Customers plan and host events seamlessly via an online platform.  
   
As a part of this requirement, Nikhil wanted to write a piece of code for his company’s Amphi Event Management System that will display customized welcome messages by taking Customers’ name as input. Help Nikhil on the task.  
   
**Input Format:**  
First line of the input is a string that corresponds to a Customer’s name. Assume that the maximum length of the string is 50.  
  
**Output Format:**  
Output should display the welcome message along with the Customer’s name.  
Refer sample input and output for formatting specifications.  
**[All text in bold corresponds to input and rest corresponds to output.]**  
  
**Sample Input and Output:**  
Enter your name  
**Beena**  
Hello Beena ! Welcome to Amphi Event Management System

import java.util.Scanner;

class Main {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

System.out.println("Enter your name");

String name=sc.nextLine();

int n=name.length();

if(n<=50)

{

System.out.println("Hello "+name+" ! Welcome to Amphi Event Management System");

}

}

}

**Total Expenses for the Event**

The prime functionality of an Event Management System is budgeting. An Event Management System should estimate the total expenses incurred by an event and the percentage rate of each of the expenses involved in planning and executing an event. Nikhil, the founder of "Pine Tree" wanted to include this functionality in his company’s Amphi Event Management System and requested your help in writing a program for the same.  
   
The program should get the branding expenses, travel expenses, food expenses and logistics expenses as input from the user and calculate the total expenses for an event and the percentage rate of each of these expenses.  
   
**Input Format:**  
First input is a int value that corresponds to the branding expenses.  
Second input is a int value that corresponds to the travel expenses.  
Third input is a int value that corresponds to the food expenses.  
Fourth input is a int value that corresponds to the logistics expenses.  
   
**Output Format:**  
First line of the output should display the int value that corresponds to the total expenses for the Event.  
Next four lines should display the percentage rate of each of the expenses.  
Refer sample input and output for formatting specifications.  
**[All text in bold corresponds to input and rest corresponds to output.]**  
  
**Sample Input and Output:**  
Enter branding expenses  
**20000**  
Enter travel expenses  
**40000**  
Enter food expenses  
**15000**  
Enter logistics expenses  
**25000**  
Total expenses : Rs.100000.00  
Branding expenses percentage : 20.00%  
Travel expenses percentage : 40.00%  
Food expenses percentage : 15.00%  
Logistics expenses percentage : 25.00%

Top of Form

Bottom of Form

**Additional Sample TestCases**

**Sample Input and Output 1 :**

Enter branding expenses

855

Enter travel expenses

877779

Enter food expenses

5544

Enter logistics expenses

2256

Total expenses : Rs.886434.00

Branding expenses percentage : 0.10%

Travel expenses percentage : 99.02%

Food expenses percentage : 0.63%

Logistics expenses percentage : 0.25%

import java.util.Scanner;

import java.text.DecimalFormat;

class Main {

public static void main(String[] args) {

DecimalFormat df = new DecimalFormat("0.00");

Scanner sc = new Scanner(System.in);

System.out.println("Enter branding expenses");

int branding = sc.nextInt();

System.out.println("Enter travel expenses");

int travel = sc.nextInt();

System.out.println("Enter food expenses");

int food = sc.nextInt();

System.out.println("Enter logistics expenses");

int logistics = sc.nextInt();

double totalexpense = branding+food+travel+logistics;

double brandingper=branding\*100/totalexpense;

double travelingper=travel\*100/totalexpense;

double foodper=food\*100/totalexpense;

double logisticsper=logistics\*100/totalexpense;

System.out.println("Total expenses : Rs."+df.format(totalexpense));

System.out.println("Branding expenses percentage : "+df.format(brandingper)+"%");

//System.out.print("% \n");

System.out.println("Travel expenses percentage : " +df.format(travelingper)+"%");

//cimalFormat df1 = new DecimalFormat("#.##");

System.out.println("Food expenses percentage : "+df.format(foodper)+"%");

System.out.println("Logistics expenses percentage : " +df.format(logisticsper)+"%");

}

}

**Thrill ride**

"Fantasy Kingdom" is a brand new Amusement park that is going to be inaugurated shortly in the City and is promoted as the place for breath-taking charm. The theme park has more than 30 exhilarating and thrilling rides and as a special feature of the park, the park Authorities have placed many Booking Kiosks at the entrance which would facilitate the public to purchase their entrance tickets and ride tickets.  
   
There are few rides in the park which are not suitable for Children and aged people, hence the park Authorities wanted to program the kiosks to issue the tickets based on people’s age. If the age given is less than 15 (Children) or greater than 60 (Aged), then the system should display as "Not Allowed", otherwise it should display as "Allowed". Write a block of code to help the Authorities program this functionality.  
   
**Input Format:**  
First line of the input is an integer that corresponds to the age of the person opting for the ride.  
  
**Output Format:**  
Output should display "Allowed" or "Not Allowed" based on the conditions given.  
Refer sample input and output for formatting specifications.  
  
**Sample Input 1:**  
20  
  
**Sample Output 1:**  
Allowed  
  
**Sample Input 2:**  
12  
  
**Sample Output 2:**  
Not Allowed

import java.util.\*;

import java.io.\*;

class Main{

public static void main(String[] args) throws Exception{

Scanner sc=new Scanner(System.in);

int age=sc.nextInt();

if((age<15)||(age>60)){

System.out.println("Not Allowed");

}else{

System.out.println("Allowed");}

}

}

## PROBLEM

**Character Pattern 3**

Write a program to generate a rectangular pattern of stars.

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

**Input and Output Format:**

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

**Sample Input 1:**

5

**Sample Output 1:**

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

import java.util.\*;

import java.io.\*;

class Main{

public static void main(String[] args) throws Exception{

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

for(int i=1;i<=n;i++){

for(int j=1;j<=i;j++){

System.out.print("\*");

}

System.out.println();

}

}

**Aayush's Scholarship**

Aayush studies in Teswan National University. Now is the time for exam results. Aayush similar to other students, hopes that his scores in 5 subjects in the exam could fetch him a scholarship for his GRE preparation.  
   
The following simple rules  are used to find whether he is eligible to receive scholarship:

* University follows **5** point grading system. In an exam, a student can receive any score from 2 to 5.  2 is called an F grade, meaning that student has failed that exam.
* Student should not have fail any of the exams.
* Student must obtain a full score in some of his/her exams to show that he/she is excellent in some of the subjects.
* He/She must have a grade point average not less than **4.0**

​You are given information regarding how Aayush performed in those 5 subjects . Help him determine whether he will receive the scholarship or not.  
   
**Input Format:**  
The input contains 5 integers denoting Aayush’s 5 subjects score in the exam.  
   
**Output Format:**  
Output a single line - "Yes" (without quotes) if Aayush will receive scholarship, or "No" (without quotes) otherwise.  
Refer sample input and output for formatting specifications.  
  
**Sample Input 1:**  
Enter the subject1 mark  
**3**  
Enter the subject2 mark  
**5**  
Enter the subject3 mark  
**4**  
Enter the subject4 mark  
**4**  
Enter the subject5 mark  
**3**  
  
**Sample Output 1:**  
No  
  
**Sample Input 2:**  
Enter the subject1 mark  
**3**  
Enter the subject2 mark  
**4**  
Enter the subject3 mark  
**4**  
Enter the subject4 mark  
**4**  
Enter the subject5 mark  
**5**  
  
**Sample Output 2:**  
Yes

import java.util.\*;

import java.io.\*;

class Main {

public static void main(String[] args) {

int sub1, sub2, sub3, sub4, sub5;

Scanner scan = new Scanner(System.in);

System.out.println("Enter the subject1 mark");

sub1 = scan.nextInt();

System.out.println("Enter the subject2 mark");

sub2 = scan.nextInt();

System.out.println("Enter the subject3 mark");

sub3 = scan.nextInt();

System.out.println("Enter the subject4 mark");

sub4 = scan.nextInt();

System.out.println("Enter the subject5 mark");

sub5 = scan.nextInt();

if(sub1==5||sub2==5||sub3==5||sub4==5||sub5==5)

{

if(sub1==2||sub2==2||sub3==2||sub4==2||sub5==2)

System.out.println("No");

else if((sub1+sub2+sub3+sub4+sub5)/5.0>=4.0)

System.out.println("Yes");

else

System.out.println("No");

}

else

System.out.println("No");

}

}

**Series 1**

The Event Organizing Company "Buzzcraft" focuses event management in a way that creates a win-win situation for all involved stakeholders. Buzzcraft don't look at building one time associations with clients, instead, aim at creating long-lasting collaborations that will span years to come. This goal of the company has helped them to evolve and gain more clients within notable time.  
The number of clients of the company from the start day of their journey till now is recorded sensibly and is seemed to have followed a specific series like: 2,3,5,7,11,13,17,19, 23 ...  
   
Write a program which takes an integer N as the input and will output the series till the Nth term.  
   
**Input Format:**  
First line of the input is an integer N.  
  
**Output Format:**  
Output a single line the series till Nth term, each separated by a comma.  
Refer sample input and output for formatting specifications.  
  
**Sample Input 1:**  
5  
  
**Sample Output 1:**  
2 3 5 7 11  
  
**Sample Input 2:**  
10  
  
**Sample Output 2:**  
2 3 5 7 11 13 17 19 23 29

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

int a = Integer.parseInt(s.nextLine());

int ct=0,n=0,i=1,j=1;

while(n<a) {

j=1;

ct=0;

while(j<=i) {

if(i%j==0){

ct++;

}

j++;

}

if(ct==2) {

System.out.printf("%d ",i);

n++;

}

i++;

}

}

}Top of Form

Bottom of Form

Top of Form

**Welcome Message**

"Pine Tree" is a recently launched startup Event Management company. The company gained a good reputation within a short span because of its highly reliable service delivery.  
   
Nikhil, the founder of this company wished to take the company’s services to the next step and decided to design an Event Management System that would let its Customers plan and host events seamlessly via an online platform. As a part of this requirement, Nikhil wanted to write a piece of code for his company’s Amphi Event Management System that will welcome all the Customers who are using it. Help Nikhil on the task.  
  
**Output Format:**  
Output should display "Welcome to Amphi Event Management System".  
Refer sample output for formatting specifications.  
  
**Sample Output:**  
Welcome to Amphi Event Management System

 public class Main{

public static void main(String[] args){

System.out.println("Welcome to Amphi Event Management System");

}

}

**Ticket type**

"FantasyKingdom" is a brand new Amusement park that is going to be inaugurated shortly in the City and is promoted as the place for breath-taking charm. The theme park has more than 30 exhilarating and craziest rides and as a special feature of the park, the park Authorities has placed many Ticketing Kiosks at the entrance which would facilitate the public to purchase their entrance tickets and ride tickets.  
   
The Entrance Tickets are to be issued typically based on age, as there are different fare for different age groups. There are 2 types of tickets – Child ticket and Adult ticket. If the age given is less than 15, then Child ticket is issued whereas for age greater than equal to 15, Adult ticket is issued. Write a piece of code to program this requirement in the ticketing kiosks.  
   
**Input Format:**  
First line of the input is an integer that corresponds to the age of the person.  
  
**Output Format:**  
Output should display "Child Ticket" or "Adult Ticket" based on the conditions given.  
Refer sample input and output for formatting specifications.  
  
**Sample Input 1:**  
20  
  
**Sample Output 1:**  
Adult Ticket  
  
**Sample Input 2:**  
12  
  
**Sample Output 2:**  
Child Ticket

import java.util.Scanner;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int age=sc.nextInt();

if(age>=15)

{

System.out.println("Adult Ticket");

}

else

{

System.out.println("Child Ticket");

}

}

}

**Lucky Pairs**

Richie and Riya are participating in a game called "Lucky pairs" at the Annual Game Fair in their Company. As per the rules of the contest, two members form a team and Richie initially has the number **A** and Riya has the number **B**.  
There are a total of **N** turns in the game, and Richie and Riya alternatively take turns. In each turn the player whose turn it is, multiplies his or her number by 2. Richie has the first turn. Suppose after the entire N turns, Richie’s number has become C and Riya’s number has become D. The final score of the team will be the sum of the scores (C+D) of both the players after N turns.  
   
Write a program to facilitate the quiz organizers to find the final scores of the teams.  
   
**Input Format:**  
The only line of input contains 3 integers **A**, **B**, and **N**.  
  
**Output Format:**  
Output a single line which contains the integer that gives the final score of the team which will be the sum of the scores of both the players after N turns.  
Refer sample input and output for formatting specifications.  
  
**Sample Input 1:**  
1 2 1  
  
**Sample Output 1:**  
4  
  
**Sample Input 2:**  
3 2 3  
  
**Sample Output 2:**  
16

SOLUTION

import java.util.Scanner;

public class Main{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

int A = sc.nextInt();

int B = sc.nextInt();

int N = sc.nextInt();

int sum = 0;

for(int i=1; i<=N; i++){

if(i%2 == 0)

{

B = B \* 2;

}

else{

A = A \* 2;

}

sum = A+B;

}

System.out.println(sum);

}

}

**Display Item Type**

The International Film Festival of India (IFFI), founded in 1952, is one of the most significant film festivals in Asia. The festival is for a weel and arrangements have to be made for food, chairs, tables, etc. The organizing committee plans to deposit the advance amount to the contractors on conformation of boking.  
Help them to store these details and print them in detailed view.  
  
Write a Java program to get item type, cost per day and deposit amount from user and display these details in a detailed view using the following classes and methods.

**[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 **ItemType**with the following private member variables / attributes.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| String | name |
| double | costPerDay |
| double | deposit |

Include appropriate **getters and setters.**  
  
In the **ItemType**class include the following methods.

|  |  |
| --- | --- |
| **Method** | **Description** |
| void display( ) | In this method, display the details of the ItemType in the format shown in the sample output. Include the statement ‘Item type details’ inside this method |

Create an another class **Main**and write a main method to test the above class.  
  
In the main( ) method, read the item type details from the user and call the display( ) method.  
  
**Example of getters and setters**  
  
private String name;  
  
public String getName( ) {  
        return name;  
}  
  
public void setName(String name) {  
        this.name = name;  
}  
  
private double costPerDay;  
  
public double getCostPerDay( ) {  
        return name;  
}  
  
public void setCostPerDay(double costPerDay) {  
        this.costPerDay = costPerDay;  
}  
  
private double deposit;  
  
public double getDeposit( ) {  
        return name;  
}  
  
public void setDeposit(double deposit) {  
        this.deposit = deposit;  
}  
**Input and Output Format:**  
Refer sample input and output for formatting specifications.  
Cost per day and Deposit value should be displayed upto 2 decimal places.  
**All text in bold correspondstoinput and the rest corresponds to output.**  
  
**Sample Input and Output 1:**  
Enter the item type name  
**Catering**  
Enter the cost per day  
**25000.00**  
Enter the deposit  
**10000.50**  
Item type details  
Name : Catering  
CostPerDay : 25000.00  
Deposit : 10000.50

## PROBLEM

Little App helps you discover great places to eat around or de-stress in all major cities across 20000+ merchants. Explore restaurants, spa & salons and activities to find your next fantastic deal. The development team of Little App seeks your help to find the duplication of user accounts.   
  
Write a Java program to get two users details and display whether their phone numbers are same or not with the following class and methods.  
  
**[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 **User** with the following private attributes/variables.

|  |  |
| --- | --- |
| **Date Type** | **Variable** |
| String | name |
| String | username |
| String | password |
| long | phoneNo |

Include appropriate getters and setters.  
Include four-argument  constructor with parameters in the following order,  
**public User(String name, String username, String password, long phoneNo)**  
  
Include the following method in **User** class.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public boolean comparePhoneNumber(User user) | In this method, compare the phone number of the two user and return true if both the numbers are equal else return false |

Create another class **Main**and write a main method to test the above class.  
  
**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/Output 1**  
Enter Name  
**john**  
Enter UserName  
**john@123**  
Enter Password  
**john@123**  
Enter PhoneNo  
**9092314562**  
Enter Name  
**john**  
Enter UserName  
**john@12**  
Enter Password  
**john@12**  
Enter PhoneNo  
**9092314562**  
Same Users  
  
**Sample Input/Output 2**  
Enter Name  
**ram**  
Enter UserName  
**ram####**  
Enter Password  
**ram**  
Enter PhoneNo  
**9092314562**  
Enter Name  
**john**  
Enter UserName  
**john@123**  
Enter Password  
**john@123**  
Enter PhoneNo  
**9092312102**  
Different Users

User.java

public class User {

//Fill your code

String name;

String username;

String password;

long phoneNo;

public User(){}

public User(String name, String username, String password, long phoneNo){

this.name = name;

this.username = username;

this.password = password;

this.phoneNo = phoneNo;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

//System.out.println(this.name);

}

public String getUserName() {

return username;

}

public void setUserName(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public long getPhoneNo() {

return phoneNo;

}

public void setPhoneNo(long phoneNo) {

this.phoneNo = phoneNo;

}

public boolean comparePhoneNumber(User user) {

//Fill your code

if(this.phoneNo == user.getPhoneNo()){

return true;

}

else{

return false;

}

}

}

Main.java

import java.io.\*;

//import java.util.Scanner;

//import java.text.DecimalFormat;

class Main{

public static void main(String[] args) throws IOException{

//Fill your code

//DecimalFormat df = new DecimalFormat("0.00");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String name;

System.out.println("Enter Name");

name = br.readLine();

String username;

System.out.println("Enter UserName");

username = br.readLine();

String password;

System.out.println("Enter Password");

password = br.readLine();//Double.parseDouble(br.readLine());

long phoneNo;

System.out.println("Enter PhoneNo");

phoneNo = Long.parseLong(br.readLine());

String name1;

System.out.println("Enter Name");

name1 = br.readLine();

String username1;

System.out.println("Enter UserName");

username1 = br.readLine();

String password1;

System.out.println("Enter Password");

password1 = br.readLine();//Double.parseDouble(br.readLine());

long phoneNo1;

System.out.println("Enter PhoneNo");

phoneNo1 = Long.parseLong(br.readLine());

//System.out.println("phone: " +phoneNo1);

User User = new User(name, username, password, phoneNo);

User user = new User(name1, username1, password1, phoneNo1);

boolean result = User.comparePhoneNumber(user);

if(result == true){

System.out.println("Same Users");

}

else{

System.out.println("Different Users");

}

// User.comparePhoneNumber(User);

}

}

**Rectangle Dimension Change**

Write a Java program to illustrate the method returning an objects by getting details from user and check the type of objects using instanceof and display these details in a detailed view using the following classes and methods.

**[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 **Rectangle**with the following private member variables / attributes.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| int | length |
| int | width |

Include appropriate getters and setters.

Include 2 argument constructor. The order in which the argument should be passed is**Rectangle(int length, int width)**  
  
In the **Rectangle**class include the following methods.

|  |  |
| --- | --- |
| **Method** | **Description** |
| int area( ) | This method computes the area of the rectange and returns it. |
| void display( ) | This method displays the length and width of the rectangle. Display the statement ‘Rectangle Dimension’ inside this method and also print the dimensions. |
| Rectangle dimensionChange(int d) | This method changes the rectangle dimension by increasing the length and width of the rectangle by d times. |

Create an another class **Main**and write a main() method to test the above class.  
  
In the main( ) method, read the length and width details from the user and test the above methods. Display the area of the rectange inside the main() method.  
  
**Problem Constraints:**  
1. Use **instanceof** operator to check the object returned by dimensionChange( ) method.  
**[**The java **instanceof**operator is used to test whether the object is an instance of the specified type (class or subclass or interface).**]**  
  
**Input and Output Format:**  
Refer sample input and output for formatting specifications.  
**[All text in bold correspondstoinput and the rest corresponds to output.**]  
  
**Sample Input and Output 1:**  
Enter the length of the rectangle  
**5**  
Enter the width of the rectangle  
**6**  
Rectangle Dimension  
Length:5  
Width:6  
Area of the Rectangle:30  
Enter the new dimension  
**2**  
Rectangle Dimension  
Length:10  
Width:12  
Area of the Rectangle:120

Top of Form

Bottom of Form

### Problem Requirements:

#### Java

|  |  |  |
| --- | --- | --- |
| Keyword | Min Count | Max Count |
| instanceof | 1 | - |

## PROBLEM

Write a program to implement the String methods like substring, charAt, equalsIgnoreCase and concat.

**Input format:**

Input consists of two strings. String length should be greater than 5.

**Output format:**

Output consists of the result of all String methds.

**Note:**

**Refer the sample input and output for specifications.**

**All text in bold corresponds to the input and remaining corresponds to the output.**

**Sample Input and Output**

Enter the first string :   
**Amphisoft**  
Enter the second string :   
**TECHNOLOGIES**  
Substring : hisoft  
Character at 3rd position is : H  
Is str1 and str2 equal : false  
Concatenated string : AmphisoftTECHNOLOGIES

## PROBLEM

Write a Java program to display the array of Integers and array of Strings. Use for each loop to iterate and print the elements.

**Constraints :**

Use for each loop to iterate and print the elements.

**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 numbers :   
**100  
23  
15**  
Enter strings :   
**hi  
hello  
welcome**  
Displaying numbers  
100  
23  
15  
Displaying strings  
hi  
hello  
welcome

Top of Form

Bottom of Form

### Problem Requirements:

#### Java

|  |  |  |
| --- | --- | --- |
| Keyword | Min Count | Max Count |
| for | 1 | 4 |

## PROBLEM

**Command Line Argument - Count**

Write a program to accept strings as command line argument and print the number of arguments entered.  
  
**Sample Input (Command Line Argument) 1:**  
Command Arguments  
  
**Sample Output 1:**  
Arguments :  
Command  
Arguments  
Number of arguments is 2  
  
**Sample Input (Command Line Argument) 1:**  
Commands  
  
**Sample Output 2:**  
Arguments :  
Commands  
Number of arguments is 1

**Customer Class With Constructor**

Refering to the SRS document, we were able to create classes for representing Customers and their Addresses. To populate values into the objects created by classes, one of the prefered ways is using Constructors. Constructors are member functions which are called when an object is created.

Write a program to get the customer details, assign the values to object and display it.

Create a class named **Customer** with the following public member variables

| **Data Type** | **Variable Name** |
| --- | --- |
| String | customerName |
| String | customerEmail |
| String | customerType |
| String | customerAddress |

Include 4 argument constructors in the Customer class in the following order**Customer(String customerName, String customerEmail, String customerType,String customerAddress)**  
  
Include the following method in the **Customer**class

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| void displayDetails() | To display the details ofthe customer in given format. |

Create a Main class to include the main() method.

In the main method

* Obtain the details of the customer.
* Create an object for Customer class using parameterized constructor(customerName, customerEmail, customerType, customerAddress)
* Call the method displayDetails() in Main class

**Note :**

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

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

**Sample Input and Output:**  
Enter the Customer Details  
Enter the name  
**Yogi**  
Enter the email  
**yogi@mail.com**  
Enter the type  
**Domestic**  
Enter the location  
**India**  
Name: Yogi  
E-mail: yogi@mail.com  
Type: Domestic  
Location: India

**Sum of an array**

Write a program to find the sum of the elements in an array using for each loop.

**Input Format:**

Input consists of n+1 integers. The first integer corresponds to ‘n’ , the size of the array. The next ‘n’ integers correspond to the elements in the array. Assume that the maximum value of n is 15.

**Output Format:**

Refer sample output for details.  
  
All text in bold corresponds to the input and remaining corresponds to the output.

**Sample Input and Output:**

Enter n :  
**5  
2  
3  
6  
8  
1**  
Sum of array elements is : 20

Top of Form

Bottom of Form

### Problem Requirements:

#### Java

|  |  |  |
| --- | --- | --- |
| Keyword | Min Count | Max Count |
| for | 1 | 2 |

## PROBLEM

Write a program to implement the String methods to convert given strings into uppercase and lowercase letters.

**Input format:**

Input consists of two strings.

**Output format:**

The first line of output should display the string in uppercase characters. (Convert first string)  
The second line of output should display the string in lowercase characters. (Convert second string)

**Note:**

**Refer the sample input and output for specifications.**

**All text in bold corresponds to the input and remaining corresponds to the output.**

**Sample Input and Output**

Enter the first string :   
**Amphisoft**  
Enter the second string :   
**TECHNOLOGIES**  
Upper Case : AMPHISOFT  
Lower Case : technologies

**Command Line Argument - Print String**

Write a program to accept a string as command line argument and print the same.  
  
**Sample Input (Command Line Argument) 1:**  
Programming  
  
**Sample Output 1:**  
Programming - Command Line Arguments  
  
**Sample Input (Command Line Argument) 2:**  
Arguments  
  
**Sample Output 2:**  
Arguments - Command Line Arguments

**Customer Address**

Write a program to get the address details and display it using classes and objects.

**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**

Create a class named **Address** with the following public attributes

|  |  |
| --- | --- |
| **Data Type** | **Attribute** |
| String | street |
| String | city |
| int | pincode |
| String | country |

Create a class named **Address**and include the following methods

|  |  |
| --- | --- |
| Method | Description |
| void displayAddress() | This method is used to display all the details. |

Create a Main class to include the main method and test the above class.

In the main method

* Obtain the details of the Address.
* Create an object for Address class and assign the values to the attribute
* Call the method displayAddress() in the Main class

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

Enter Customer Address  
Enter the street  
**13,Rockfort Street**  
Enter the city  
**Chennai**  
Enter the pincode  
**654035**  
Enter the country  
**India**  
Street: 13,Rockfort Street  
City: Chennai  
Pincode: 654035  
Country: India

public class Address{

private String Street;

private String City;

private int Pincode;

private String Country;

Address(String street, String city, int pincode, String country){

this.Street = street;

this.City = city;

this.Country = country;

this.Pincode = pincode;

/\*public void setStreet(String street) {

this.street = street;

}

public String getStreet() {

return street;

}

public void setCity(String city) {

this.city = city;

}

public String getCity() {

return city;

}

public void setPincode(int pincode) {

this.pincode = pincode;

}

public int getPincode() {

return pincode;

}

public void setCountry(String country) {

this.country = country;

}

public String getCountry() {

return country;

}\*/

}

void displayAddress() {

System.out.println("Street: "+this.Street);

System.out.println("City: "+this.City);

System.out.println("Pincode: "+this.Pincode);

System.out.println("Country: "+this.Country);

}

}

Main

import java.util.Scanner;

import java.io.\*;

public class Main{

public static void main(String[] args) throws IOException{

//Fill your code

int pincode;

String country, street, city;

InputStreamReader stream = new InputStreamReader(System.in);

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

//Scanner sc = new Scanner(System.in);

System.out.println("Enter Customer Address");

System.out.println("Enter the street");

street=br.readLine();

//street = sc.nextLine();

System.out.println("Enter the city");

city=br.readLine();

//city = sc.nextLine();

System.out.println("Enter the pincode");

pincode = Integer.parseInt(br.readLine());

//pincode = sc.nextInt();

System.out.println("Enter the country");

country=br.readLine();

//country = sc.nextLine();

Address obj= new Address(street, city, pincode, country);

obj.displayAddress();

}

}

**Wrapper Class – Integer I**

This program is to illustrate Integer Wrapper class.

Write a program that accepts a “**Integer**” class type value as input and invokes some of the methods defined in the Integer Wrapper class.

Refer sample input and output. All functions should be performed using the methods defined in the Integer class.

**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 an integer

**540**

The binary equivalent of 540 is 1000011100

The hexadecimal equivalent of 540 is 21c

The octal equivalent of 540 is 1034

Byte value of 540 is 28

Short value of 540 is 540

Long value of 540 is 540

Float value of 540 is 540.0

Double value of 540 is 540.0

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int num;

System.out.println("Enter an integer");

num = sc.nextInt();

System.out.println("The binary equivalent of "+num+" is "+ Integer.toBinaryString(num));

System.out.println("The hexadecimal equivalent of "+num+" is "+ Integer.toHexString(num));

System.out.println("The octal equivalent of "+num+" is "+ Integer.toOctalString(num));

byte b = (byte) num;

Short s = (short) num;

long l = num;

Float f = (float) num;

Double d = (double) num;

System.out.println("Byte value of "+num+" is "+b);

System.out.println("Short value of "+num+" is "+s);

System.out.println("Long value of "+num+" is "+l);

System.out.println("Float value of "+num+" is "+f);

System.out.println("Double value of "+num+" is "+d);

}

}

**Duplicate mobile number exception**

Write a java program to find the duplicate mobile number using the exception handling mechanism.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**  
  
Create a Class called **ContactDetail** with the following private attributes.

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| mobile | String |
| alternateMobile | String |
| landLine | String |
| email | String |
| address | String |

Include getters and setters.  
Include default and parameterized constructors.  
Format for a parameterized constructor is **ContactDetail(String mobile, String alternateMobile,String landLine, String email, String address)**  
  
Override the **toString()** method to display the Contact details as specified.  
  
Create a class called **ContactDetailBO** with following methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| static void validate(String mobile,String alternateMobile) | This method throws DuplicateMobileNumber exception if the mobile and alternateMobile are the same. |

Create a driver class called **Main**. In the Main method, obtain inputs from the user. Validate the mobile and alternateMobile and display the ContactDetail if no exception occurs else handle the exception.  
  
Pass the exception message as "**Mobile number and alternate mobile number are same**". If mobile and alternateMobile are the same.  
  
**Input and Output format:**  
Refer to sample Input and Output for formatting specifications.  
  
**Note: All text in bold corresponds to the input and rest corresponds to the output.**  
  
**Sample Input and Output 1:**  
  
Enter the contact details  
**9874563210,9874563210,0447896541,johndoe@abc.in,22nd street kk nagar chennai**  
DuplicateMobileNumberException: Mobile number and alternate mobile number are same  
  
**Sample Input and Output 2:**  
  
Enter the contact details  
**9874563210,9876543211,0447896541,johndoe@abc.in,22nd lane RR nagar kovai**  
Mobile:9874563210  
Alternate mobile:9876543211  
LandLine:0447896541  
Email:johndoe@abc.in  
Address:22nd lane RR nagar kovai

public class DuplicateMobileNumberException extends Exception{

public DuplicateMobileNumberException(String s)

{

super(s);

}

public String toString(){

return "DuplicateMobileNumberException: Mobile number and alternate mobile number are same ";

}

}

public class ContactDetailBO {

static void validate(String mobile, String alternateMobile) throws DuplicateMobileNumberException {

if (mobile.equals(alternateMobile)) {

throw new DuplicateMobileNumberException("DuplicateMobileNumberException :");

}

}

}-----------

public class ContactDetail {

//Your code here

private String mobile;

private String alternateMobile;

private String landLine;

private String email;

private String address;

public ContactDetail(String mobile, String alternateMobile, String landLine, String email, String address) {

this.mobile = mobile;

this.alternateMobile = alternateMobile;

this.landLine = landLine;

this.email = email;

this.address = address;

}

public ContactDetail()

{

}

public String getMobile() {

return mobile;

}

public void setMobile(String mobile) {

this.mobile = mobile;

}

public String getAlternateMobile() {

return alternateMobile;

}

public void setAlternateMobile(String alternateMobile) {

this.alternateMobile = alternateMobile;

}

public String getLandLine() {

return landLine;

}

public void setLandLine(String landLine) {

this.landLine = landLine;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String toString()

{

return "Mobile:"+this.mobile+"\nAlternate mobile:"+this.alternateMobile+"\nLandLine:"+this.landLine+"\nEmail:"+this.email+"\nAddress:"+this.address;

}

}

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 contact details");

String st = br.readLine();

String[] str = st.split(",");

ContactDetail userInfo = new ContactDetail(str[0], str[1], str[2], str[3], str[4]);

ContactDetailBO BO = new ContactDetailBO();

try {

BO.validate(str[0], str[1]);

System.out.println(userInfo.toString());

} catch (DuplicateMobileNumberException e) {

System.out.println(e.toString());

}

}

}

**SeatNotAvailableException**

An organization is organizing a charity fate for the well being of poor kids. Since the manager was running short on time, he asked you to help him with the ticket bookings. You being from a programming background decide to design a program that asks the user about the seat number they want. Seat booking details are stored in an array. If the seat number requested is available booking should be done else print the message " **SeatNotAvailableException**". If the seat number requested is not in the range throws an exception **ArrayIndexOutOfBoundsException.**  
  
Create a class **SeatNotAvailableException** that extends Exception.  
  
Create an array of size n\*n (n rows each with n seats) which is got from the user. Get the tickets to be booked from the user and handle any exception that occurs in **Main**Class. (Take seat numbers from 0 to (n\*n)-1)  
  
**Note**: Vacant seats are denoted by (**0)**and booked seats by (**1)**. Show message as "**Already Booked**" as a Custom exception.  
  
**Input and Output format:**  
Refer sample Input and Output for formatting specifications.t of the output.  
  
**[All Texts in bold corresponds to the input and rest are output]  
Sample Input and Output 1:**  
  
Enter the number of rows and columns of the show:  
**3**  
Enter the number of seats to be booked:  
**2**  
Enter the seat number 1  
**8**  
Enter the seat number 2  
**0**  
The seats booked are:  
1 0 0  
0 0 0  
0 0 1  
  
**Sample Input and Output 2:**  
  
Enter the number of rows and columns of the show:  
**3**  
Enter the number of seats to be booked:  
**2**  
Enter the seat number 1  
**9**  
java.lang.ArrayIndexOutOfBoundsException: 9  
The seats booked are:  
0 0 0  
0 0 0  
0 0 0  
  
**Sample Input and Output 3:**  
  
Enter the number of rows and columns of the show:  
**4**  
Enter the number of seats to be booked:  
**3**  
Enter the seat number 1  
**15**  
Enter the seat number 2  
**14**  
Enter the seat number 3  
**15**  
SeatNotAvailableException: Already Booked  
The seats booked are:  
0 0 0 0  
0 0 0 0  
0 0 0 0  
0 0 1 1

public class SeatNotAvailableException extends Exception{

public SeatNotAvailableException(String s) {

super(s);

}

}   
 ----------------------------------------------------------------------------------------------------------

import java.util.Scanner;

import java.io.\*;

public class Main {

public static void main(String args[]) throws SeatNotAvailableException{

Scanner sc = new Scanner(System.in);

System.out.println("Enter the number of rows and columns of the show:");

int n = sc.nextInt();

int size = (n\*n);

int s;

int a[] = new int[size];

int mat[][] = new int[n][n];

System.out.println("Enter the number of seats to be booked:");

int seats = sc.nextInt();

try {

for(int i=0;i<seats;i++) {

System.out.println("Enter the seat number "+(i+1));

s = sc.nextInt();

if(a[s]==0) {

a[s] =1;

int t=0;

for(int j=0;j<n;j++) {

for(int k=0;k<n;k++) {

mat[j][k] = a[t];

t++;

}

}

}else {

throw new SeatNotAvailableException("Already Booked");

}

}

}catch (Exception e) {

System.out.println(e);

}

finally {

System.out.println("The seats booked are:");

for(int i=0;i<n;i++) {

for(int j=0;j<n;j++) {

System.out.print(mat[i][j]+" ");

}

System.out.println();

}

}

}

}

**Parse Exception**

For our application, we would have obtained date inputs. If the user enters a different format other than specified, an **Invalid Date Exception** occurs and the program is interrupted. To avoid that, handle the exception and prompt the user to enter the right format as specified.  
  
Create a driver class called **Main**. In the main method, Obtain start time and end time for stage event show, if an exception occurs, handle the exception and notify the user about the right format.  
  
**Input format:**  
The input consists of the start date and end date.   
The format for the date is **dd-MM-yyyy-HH:mm:ss**  
  
**Output format:**  
Refer sample Input and Output for formatting specifications

**Note: All text in bold corresponds to the input and rest corresponds to the output.**  
  
**Sample Input and Output 1:**  
  
Enter the stage event start date and end date  
**27-01-2017-12**  
Input dates should be in the format 'dd-MM-yyyy-HH:mm:ss'  
  
**Sample Input and Output 2:**  
  
Enter the stage event start date and end date  
**27-01-2017-12:0:0  
28-01-2017-12:0:0**  
Start date:27-01-2017-12:00:00  
End date:28-01-2017-12:00:00

import java.util.\*;

import java.text.ParseException;

import java.text.SimpleDateFormat;

public class Main {

public static void main(String[] args) {

// your code here

String d,d1;

Scanner s=new Scanner(System.in);

try{

System.out.println("Enter the stage event start date and end date");

d=s.next();

SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy-HH:mm:ss");

String d2=sdf.format(sdf.parse(d));

d1 =s.next();

String d3=sdf.format(sdf.parse(d1));

System.out.println("Start date:"+d2);

System.out.println("End date:"+d3);

}catch (ParseException e) {

System.out.println("Input dates should be in the format 'dd-MM-yyyy-HH:mm:ss'");

}

}

**Arithmetic Exception**

 An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at runtime, it disrupts the normal flow of the program. For example, there are 10 statements in your program and there occurs an exception at statement 5, the rest of the code will not be executed i.e. statement 6 to 10 will not run. If we perform exception handling, the rest of the statement will be executed. That is why we use exception handling.

For practice in exception handling, obtain the cost for 'n' days of an item and n as input and calculate the cost per day for the item. In case, zero is given as input for n, an arithmetic exception is thrown, handle the exception and prompt the user accordingly.

Create a driver class called **Main**. In the Main method, obtain input from the user and store the values in int type. Handle exception if one occurs.  
  
**Input format:**  
The first line of input is an integer which corresponds to the cost of the item for n days.  
The second line of input is an integer which corresponds to the value n.  
  
**Output format:**  
If the value of n is zero throws an exception.  
Otherwise, print the integer output which corresponds to the cost per day of the item.

**NOTE: All text in bold corresponds to the input and rest corresponds to the output.**  
  
**Sample Input  and Output 1:**  
  
Enter the cost of the item for n days  
**100**  
Enter the value of n  
**0**  
java.lang.ArithmeticException: / by zero  
  
**Sample Input and Output 2:**  
  
Enter the cost of the item for n days  
**100**  
Enter the value of n  
**20**  
Cost per day of the item is 5

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the cost of the item for n days");

int days = sc.nextInt();

System.out.println("Enter the value of n");

int num = sc.nextInt();

try{

int cost = days/num;

System.out.println("Cost per day of the item is "+cost);

}catch(ArithmeticException e){

System.out.println(e);

}

}

}

**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 seats in the event and the user tries to book the 105th seat, it will throw this exception. 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  
**100**  
The seats booked are:  
23  
42  
65  
81  
100  
  
**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**  
java.lang.ArrayIndexOutOfBoundsException: 100

import java.util.Scanner;

public class Main {

public static void main(String args[]) {

int[] seatArray = new int[100];

int n,count = 1;

Scanner sc = new Scanner(System.in);

System.out.println("Enter the number of seats to be booked:");

n = sc.nextInt();

try{

for(int i=0;i<n;i++) {

System.out.println("Enter the seat number "+(i+1));

count = sc.nextInt();

seatArray[count-1] = count;

}

System.out.println("The seats booked are:");

for(int i = 0; i< 100; i++){

if(seatArray[i] != 0){

System.out.println(seatArray[i]);

}

}

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println(e);

}

}

}

**Single inheritance**

Write a Java program to implement Single Inheritance.  
  
**[Note : Strictly adhere to the object oriented specifications given as a part of the problem statement. Use the same class names and member variable names.  
Follow the naming conventions mentioned for getters / setters. Create separate classes in separate files.** **Do not create the classes within namespaces]**   
  
Create a class named **Person**with the following private data members.

|  |  |
| --- | --- |
| **Data Type** | **Data Member** |
| String | name |
| String | dateOfBirth |
| String | gender |
| String | mobileNumber |
| String | bloodGroup |

Include appropriate getters and setters.  
  
Create a class named **Donor**which extends **Person**class with the following private data members.

|  |  |
| --- | --- |
| **Data Type** | **Data Member** |
| String | bloodBankName |
| String | donorType |
| String | donationDate |

Include appropriate getters and setters.  
  
The class **Donor**should have the following method

|  |  |
| --- | --- |
| **Method** | **Member Function** |
| public void displayDonationDetails( ) | This method displays the donation details. Display the statement ‘Donation Details :’ inside this method |

Create an another class **Main**and write a main method to test the above class.  
  
In the main( ) method, read the person and donor details from the user and call the displayDonationDetails( ) method.  
  
**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 the name :

**Justin**

Enter Date of Birth :

**11-01-1995**

Enter Gender :

**Male**

Enter Mobile Number :

**9994910354**

Enter Blood Group :

**B+ve**

Enter Blood Bank Name :  
**Blood Assurance**

Enter Donor Type :

**Whole Blood**

Enter Donation Date :

**09-07-2017**  
Donation Details :

Name : Justin

Date Of Birth : 11-01-1995

Gender : Male

Mobile Number : 9994910354

Blood Group : B+ve

Blood Bank Name : Blood Assurance

Donor Type : Whole Blood

Donation Date : 09-07-2017

class Donor extends Person{

private String bloodBankName;

private String donorType;

private String donationDate;

public Donor() {

}

public String getBloodBankName() {

return bloodBankName;

}

public void setBloodBankName(String bloodBankName) {

this.bloodBankName = bloodBankName;

}

public String getDonorType() {

return donorType;

}

public void setDonorType(String donorType) {

this.donorType = donorType;

}

public String getDonationDate() {

return donationDate;

}

public void setDonationDate(String donationDate) {

this.donationDate = donationDate;

}

public void displayDonationDetails( ) {

System.out.println("Donation Details :");

System.out.println("Name : "+getName());

System.out.println("Date Of Birth : "+getDateOfBirth());

System.out.println("Gender : "+getGender());

System.out.println("Mobile Number : "+getMobileNumber());

System.out.println("Blood Group : "+getBloodGroup());

System.out.println("Blood Bank Name : "+getBloodBankName());

System.out.println("Donor Type : "+getDonorType());

System.out.println("Donation Date : "+getDonationDate());

//Fill your code here

}

}

class Person{

private String name;

private String dateOfBirth;

private String gender;

private String mobileNumber;

private String bloodGroup;

Person(){

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDateOfBirth() {

return dateOfBirth;

}

public void setDateOfBirth(String dateOfBirth) {

this.dateOfBirth = dateOfBirth;

}

public String getGender() {

return gender;

}

public void setGender(String gender) {

this.gender = gender;

}

public String getMobileNumber() {

return mobileNumber;

}

public void setMobileNumber(String mobileNumber) {

this.mobileNumber = mobileNumber;

}

public String getBloodGroup() {

return bloodGroup;

}

public void setBloodGroup(String bloodGroup) {

this.bloodGroup = bloodGroup;

}

}

import java.util.Scanner;

public class Main {

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

Donor d = new Donor();

System.out.println("Enter the name :");

d.setName(sc.nextLine());

System.out.println("Enter Date of Birth :");

d.setDateOfBirth(sc.nextLine());

System.out.println("Enter Gender :");

d.setGender(sc.nextLine());

System.out.println("Enter Mobile Number :");

d.setMobileNumber(sc.nextLine());

System.out.println("Enter Blood Group :");

d.setBloodGroup(sc.nextLine());

System.out.println("Enter Blood Bank Name :");

d.setBloodBankName(sc.nextLine());

System.out.println("Enter Donor Type :");

d.setDonorType(sc.nextLine());

System.out.println("Enter Donation Date :");

d.setDonationDate(sc.nextLine());

d.displayDonationDetails();

//Fill your code here

}

}

**Calculate Reward Points**

**[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.]**  
  
ABC Bank announced a new scheme of reward points for a transaction using an ATM card. Each transaction using the normal card will be provided by 1% of the transaction amount as reward point. If a transaction is made using a premium card and it is for fuel expenses, additional 10 points will be rewarded. Help the bank to calculate the total reward points.  
  
Create a class **VISACard**with the following method.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public double computeRewardPoints(String type, double amount) | This method returns the 1% of the transaction amount as reward points |

Create a class **HPVISACard**which extends **VISACard** class and overrides the following method.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public double computeRewardPoints(String type, double amount) | In this method, calculate the reward points from the base class and add 10 points if it is for fuel expense |

**Hint:**  
Use super keyword to calculate reward points from base class.  
  
Create **Main**class with **main** method, get the transaction details as a comma separated values.  
**(Transaction type, amount, card type)**  
  
Card type will be either **‘VISA card’ or ‘HPVISA card’**. Otherwise, display **‘Invalid data’**  
  
Calculate the reward points corresponding to the card type and transaction type and print the reward points(upto two precision).  
  
**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 transaction detail  
**Shopping,5000,VISA card**  
Total reward points earned in this transaction is 50.00  
Do you want to continue?(Yes/No)  
**Yes**  
Enter the transaction detail  
**Fuel,5000,HIVISA card**  
Invalid data  
Do you want to continue?(Yes/No)  
**Yes**  
Enter the transaction detail  
**Fuel,5000,HPVISA card**  
Total reward points earned in this transaction is 60.00  
Do you want to continue?(Yes/No)  
**No**

Top of Form

Bottom of Form

public class VISACard {

public double computeRewardPoints(String type, double amount) {

return amount \* 0.01;

}

}

class HPVISACard extends VISACard{

public double computeRewardPoints(String type, double amount){

double total=super.computeRewardPoints(type, amount);

if(type.equalsIgnoreCase("Fuel")) {

total=total+10;

return total;

}else{

return total;

}

}

}

import java.io.\*;

import java.util.\*;

import java.text.DecimalFormat;

public class Main {

public static void main(String[] args) {

String tDetails = null;

String transactionType = null;

double amount = 0;

String cardType = null;

String flag = null;

VISACard v = new VISACard();

HPVISACard h = new HPVISACard();

Scanner sc = new Scanner(System.in);

do {

System.out.println("Enter the transaction detail");

tDetails = sc.nextLine();

String dSplit[] = tDetails.split(",");

transactionType = dSplit[0];

amount = Double.parseDouble(dSplit[1]);

cardType = dSplit[2];

DecimalFormat df = new DecimalFormat("0.00");

if (cardType.equals("VISA card")) {

System.out.println("Total reward points earned in this transaction is "

+ df.format(v.computeRewardPoints(transactionType, amount)));

System.out.println("Do you want to continue?(Yes/No)");

flag = sc.nextLine();

} else if (cardType.equals("HPVISA card")) {

System.out.println("Total reward points earned in this transaction is "

+ df.format(h.computeRewardPoints(transactionType, amount)));

System.out.println("Do you want to continue?(Yes/No)");

flag = sc.nextLine();

} else {

System.out.println("Invalid data");

System.out.println("Do you want to continue?(Yes/No)");

flag = sc.nextLine();

}

} while (flag.equals("Yes"));

}

}

**GST Calculation**

**[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.]**  
  
Write a program to calculate the total amount with GST for the events. There are two types of Events Stage show and Exhibition. For Stage show GST will be 15% and for exhibition GST will be 5%  
  
Create class **Event** with the following protected attributes/variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| String | name |
| String | type |
| double | costPerDay |
| int | noOfDays |

Include a four argument constructor with parameters in the following order,  
**public Event(String name, String type, double costPerDay, int noOfDays)**  
  
Create class **Exhibition** which extends the **Event** class with the following private attributes/variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| static int | gst = 5 |
| int | noOfStalls |

Include a five argument constructor with parameters in the following order,  
**public Exhibition(String name, String type, double costPerDay, int noOfDays, int noOfStalls)**  
  
Include the following methods.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public double totalCost() | This method is to calculate the total amount with 5% GST |

Create class **StageEvent** which extends the **Event** class with the following private attributes/variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| static int | gst = 15 |
| int | noOfSeats |

Include a five argument constructor with parameters in the following order,  
**public StageEvent(String name, String type, double costPerDay, int noOfDays, int noOfSeats)**  
  
Include the following method.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public double totalCost() | This method is to calculate the total amount with 15% GST |

Use super( ) to call and assign values in base class constructor.  
Override toString() method to display the event details. Refer the sample output format.  
  
Create **Main** class with **main** method.  
In the main() method, read the event details from the user and then create the object of the event according to the event type.  
The total amount will be calculated according to the GST of the corresponding event. Display the total amount upto two precision.  
  
**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:**  
Enter event name  
**Sky Lantern Festival**  
Enter the cost per day  
**1500**  
Enter the number of days  
**3**  
Enter the type of event  
1.Exhibition  
2.Stage Event  
**2**  
Enter the number of seats  
**100**  
Event Details  
Name:Sky Lantern Festival  
Type:Stage Event  
Number of seats:100  
Total amount: 5175.00  
  
**Sample Input and Output 2:**  
Enter event name  
**Glastonbury**  
Enter the cost per day  
**5000**  
Enter the number of days  
**2**  
Enter the type of event  
1.Exhibition  
2.Stage Event  
**1**  
Enter the number of stalls  
**10**  
Event Details  
Name:Glastonbury  
Type:Exhibition  
Number of stalls:10  
Total amount: 10500.00  
  
**Sample Input and Output 3:**  
Enter event name  
**Glastonbury**  
Enter the cost per day  
**5000**  
Enter the number of days  
**2**  
Enter the type of event  
1.Exhibition  
2.Stage Event  
**3**  
Invalid input

import java.util.\*;

import java.text.DecimalFormat;

public class Main {

public static void main(String[] args) {

String name, type;

double cpd;

int npd;

Scanner sc = new Scanner(System.in);

System.out.println("Enter event name");

name = sc.nextLine();

System.out.println("Enter the cost per day");

cpd = sc.nextDouble();

System.out.println("Enter the number of days");

npd = sc.nextInt();

sc.nextLine();

System.out.println("Enter the type of event\n1.Exhibition\n2.Stage Event");

type = sc.nextLine();

DecimalFormat df = new DecimalFormat("0.00");

switch (type) {

case "1":

System.out.println("Enter the number of stalls");

int stalls = sc.nextInt();

Exhibition e = new Exhibition(name, type, cpd, npd, stalls);

Double total = e.totalCost();

System.out.println(e.toString() + "\nTotal amount:" + df.format(total));

break;

case "2":

System.out.println("Enter the number of seats");

int seats = sc.nextInt();

StageEvent s = new StageEvent(name, type, cpd, npd, seats);

Double total1 = s.totalCost();

System.out.println(s.toString() + "\nTotal amount:" + df.format(total1));

break;

default:

System.out.println("Invalid input");

break;

}

}

}

class Event{

protected String name;

protected String type;

protected double costPerDay;

protected int noOfDays;

public Event(String name, String type, double costPerDay, int noOfDays) {

super();

this.name = name;

this.type = type;

this.costPerDay = costPerDay;

this.noOfDays = noOfDays;

}

}

class StageEvent extends Event {

static int gst = 15;

int noOfSeats;

public StageEvent(String name, String type, double costPerDay, int noOfDays, int noOfSeats) {

super(name, type, costPerDay, noOfDays);

this.noOfSeats = noOfSeats;

}

public double totalCost() {

double totalAmount = (costPerDay \* noOfDays);

double gstAmount = (totalAmount \* gst) / 100;

double total = (totalAmount + gstAmount);

return total;

}

@Override

public String toString() {

return "Event Details\nName:" + name + "\nType:Stage Event\nNumber of seats:" + noOfSeats;

}

}

class Exhibition extends Event {

static int gst = 5;

int noOfStalls;

public Exhibition(String name, String type, double costPerDay, int noOfDays, int noOfStalls) {

super(name, type, costPerDay, noOfDays);

this.noOfStalls = noOfStalls;

}

public double totalCost() {

double totalAmount = (costPerDay \* noOfDays);

double gstAmount = (totalAmount \* gst) / 100;

double total = (totalAmount + gstAmount);

return total;

}

@Override

public String toString() {

return "Event Details\nName:" + name + "\nType:Exhibition\nNumber of stalls:" + noOfStalls;

}

}

**Abstract Event**

Let's have a practice in creating an Abstract class for the Event. In this application create an abstract class Event, StageEvent class and a class Exhibition with the provided attributes and let's implement an abstract method to calculate the total cost of the event and print the details of the particular event of this application.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**  
  
Create an **abstract** class called **Event** with following protected attributes.

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| name | String |
| detail | String |
| type | String |
| organiser | String |

Include 4 argument constructors in the Event class in the following order **Event(String name, String detail, String type, String organiser)**

Include the following abstract method in the class Event.

|  |  |
| --- | --- |
| **Method** | **Description** |
| abstract Double calculateAmount() | an abstract method |

Create a class named **Exhibition**which extends **Event** class with the following private attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| noOfStalls | Integer |
| rentPerStall | Double |

Include a six argument constructor with parameters in the following order,  
**public Exhibition(String name, String detail, String type, String organiser, Integer noOfStalls,Double rentPerStall)**

Use super( ) to call and assign values in base class constructor.

Include the following abstract method in the class Exhibition.

|  |  |
| --- | --- |
| **Method** | **Description** |
| Double calculateAmount () | This method returns the product of noOfStalls and rentPerStall |

Create a class named **StageEvent** which extends **Event**class with the following private attributes.

|  |  |
| --- | --- |
| **Attribute** | **Datatype** |
| noOfShows | Integer |
| costPerShow | Double |

Include a six argument constructor with parameters in the following order,  
**public StageEvent(String name, String detail, String type, String organiser, Integer noOfShows,Double costPerShow)**

Use super( ) to call and assign values in base class constructor.

Include the following abstract method in the class StageEvent.

|  |  |
| --- | --- |
| **Method** | **Description** |
| Double calculateAmount() | This method returns the product of noOfShows and costPerShow |

Create a driver class called **Main**. In the main method, obtain input from the user and create objects accordingly.  
  
**Input format:**  
Input format for Exhibition is in the CSV format (**name,detail,type,organiser,noOfStalls,rentPerStall)**  
Input format for StageEvent is in the CSV format (**name,detail,type,organiser,noOfShows,costPerShow)**      
  
**Output format:**  
Print "**Invalid choice**" if the input is invalid to our application and terminate.  
Display one digit after the decimal point for Double datatype.  
Refer to sample Input and Output for formatting specifications.  
  
**Note: All text in bold corresponds to the input and rest corresponds to output.**  
  
**Sample Input and output 1:**

Enter your choice

1.Exhibition

2.StageEvent

**1**

Enter the details in CSV format

**Book expo,Special sale,Academics,Mahesh,100,1000**

Exhibition Details

Event Name:Book expo

Detail:Special sale

Type:Academics

Organiser Name:Mahesh

Total Cost:100000.0

**Sample Input and Output 2:**

Enter your choice

1.Exhibition

2.StageEvent

**2**

Enter the details in CSV format

**JJ magic show,Comedy magic,Entertainment,Jegadeesh,5,1000**

Stage Event Details

Event Name:JJ magic show

Detail:Comedy magic

Type:Entertainment

Organiser Name:Jegadeesh

Total Cost:5000.0

**Sample Input and Output 3:**

Enter your choice

1.Exhibition

2.StageEvent

**3**

Invalid choice

Top of Form

Bottom of Form

public class Exhibition extends Event {

private Integer noOfStalls;

private Double rentPerStall;

public Exhibition(String name, String detail, String type, String organiser, Integer noOfStalls,

Double rentPerStall) {

super(name, detail, type, organiser);

this.noOfStalls = noOfStalls;

this.rentPerStall = rentPerStall;

}

Double calculateAmount() {

return noOfStalls \* rentPerStall;

}

}

public class StageEvent extends Event {

private Integer noOfShows;

private Double costPerShow;

public StageEvent(String name, String detail, String type, String organiser, Integer noOfShows,

Double costPerShow) {

super(name, detail, type, organiser);

this.noOfShows = noOfShows;

this.costPerShow = costPerShow;

}

Double calculateAmount() {

return noOfShows \* costPerShow;

}

}

public abstract class Event {

protected String name;

protected String detail;

protected String type;

protected String organiser;

public Event(String name, String detail, String type, String organiser) {

this.name = name;

this.detail = detail;

this.type = type;

this.organiser = organiser;

}

abstract Double calculateAmount();

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDetail() {

return detail;

}

public void setDetail(String detail) {

this.detail = detail;

}

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public String getOrganiser() {

return organiser;

}

public void setOrganiser(String organiser) {

this.organiser = organiser;

}

}

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s, name, detail, type, organiser;

Integer noOfStalls, noOfShows;

Double rentPerStall, costPerShow, totalCost;

System.out.println("Enter your choice");

System.out.println("1.Exhibition");

System.out.println("2.StageEvent");

int ch = sc.nextInt();

switch (ch) {

case 1:

System.out.println("Enter the details in CSV format");

sc.nextLine();

s = sc.nextLine();

String split[] = s.split(",");

name = split[0];

detail = split[1];

type = split[2];

organiser = split[3];

noOfStalls = Integer.parseInt(split[4]);

rentPerStall = Double.parseDouble(split[5]);

Exhibition e = new Exhibition(name, detail, type, organiser, noOfStalls, rentPerStall);

totalCost = e.calculateAmount();

System.out.println("Exhibition Details");

System.out.println("Event Name:" + e.getName());

System.out.println("Detail:" + e.getDetail());

System.out.println("Type:" + e.getType());

System.out.println("Organiser Name:" + e.getOrganiser());

System.out.println("Total Cost:" + totalCost);

break;

case 2:

System.out.println("Enter the details in CSV format");

sc.nextLine();

s = sc.nextLine();

String split1[] = s.split(",");

name = split1[0];

detail = split1[1];

type = split1[2];

organiser = split1[3];

noOfShows = Integer.parseInt(split1[4]);

costPerShow = Double.parseDouble(split1[5]);

StageEvent se = new StageEvent(name, detail, type, organiser, noOfShows, costPerShow);

totalCost = se.calculateAmount();

System.out.println("Stage Event Details");

System.out.println("Event Name:" + se.getName());

System.out.println("Detail:" + se.getDetail());

System.out.println("Type:" + se.getType());

System.out.println("Organiser Name:" + se.getOrganiser());

System.out.println("Total Cost:" + totalCost);

break;

default:

System.out.println("Invalid choice");

break;

}

}

}

**Overriding-simple**

Overriding is another concept that every application developer should know. Overriding is a runtime polymorphism. The inherited class has the overridden method which has the same name as the method in the parent class. The argument number, types or return types should not differ in any case. The method is invoked with the object of the specific class ( but  with the reference of the parent class).  
  
Now let's try out a simple overriding concept in our application. For this, we can take our original example of Class Event, and its child classes Exhibition and StageEvent.  
  
Create a parent class **Event** with following protected attributes,

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| name | String |
| detail | String |
| ownerName | String |

Include parameterized constructors to the Event class in the following order **Event(String name, String detail, String ownerName)**

Include the below abstract method in the Event class.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public abstract Double projectedRevenue() | Return just 0.0 |

Then create child class **Exhibition** that extends **Event** with the following attribute,

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| noOfStalls | Integer |

Include parameterized constructors to the Exhibition class in the following order **Exhibition(String name, String detail, String ownerName, Integer noOfStalls).**Use super( ) to call and assign values in base class constructor.

Add method **projectedRevenue()** in Exhibition class

|  |  |
| --- | --- |
| **Method** | **Description** |
| public Double projectedRevenue() | Calculate revenue and return the double value. Each stall will produce Rs.10000 as revenue |

And create another child class **StageEvent** that extends Event with the following attribute,

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| noOfShows | Integer |
| noOfSeatsPerShow | Integer |

Include parameterized constructors to the **StageEvent** class in the following order **StageEvent(String name, String detail, String ownerName, Integer noOfShows, Integer noOfSeatsPerShow).**Use super( ) to call and assign values in base class constructor.

Add method **projectedRevenue()** in StageEvent class

|  |  |
| --- | --- |
| **Method** | **Description** |
| public Double projectedRevenue() | Calculate revenue and return the double value. Each seat produces Rs.50 revenue. |

Refer sample input/output for other further details and format of the output.  
  
**[All Texts in bold corresponds to the input and rest are output]  
Sample Input/Output 1:**  
  
Enter the name of the event:  
**Science Fair**  
Enter the detail of the event:  
**Explore Technology**  
Enter the owner name of the event:  
**ABCD**  
Enter the type of the event:  
1.Exhibition  
2.StageEvent  
**1**  
Enter the number of stalls:  
**65**  
The projected revenue of the event is 650000.0  
  
**Sample Input/Output 2:**  
  
Enter the name of the event:  
**Magic Show**  
Enter the detail of the event:  
**See Magic without Logic**  
Enter the owner name of the event:  
**SDFG**  
Enter the type of the event:  
1.Exhibition  
2.StageEvent  
**2**  
Enter the number of shows:  
**10**  
Enter the number of seats per show:  
**100**  
The projected revenue of the event is 50000.0

public class StageEvent extends Event{

private Integer noOfShows;

private Integer noOfSeatsPerShow;

public StageEvent()

{}

public StageEvent(String name, String detail, String ownerName,Integer noOfShows,Integer noOfSeatsPerShow)

{

super(name,detail,ownerName);

this.name = name;

this.detail = detail;

this.ownerName = ownerName;

this.noOfShows = noOfShows;

this.noOfSeatsPerShow = noOfSeatsPerShow;

}

@Override

public double projectedRevenue()

{

double revenue = noOfShows \* noOfSeatsPerShow \* 50;

return revenue;

}

}

public abstract class Event {

protected String name;

protected String detail;

protected String ownerName;

public Event(){

}

public Event(String name, String detail, String ownerName)

{

this.name = name;

this.detail = detail;

this.ownerName = ownerName;

}

public double projectedRevenue(){

return 0.0;

}

}

import java.io.IOException;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws IOException {

Scanner sc = new Scanner(System.in);

double totalrevenue = 0;

System.out.println("Enter the name of the event:");

String name = sc.nextLine();

System.out.println("Enter the detail of the event:");

String detail = sc.nextLine();

System.out.println("Enter the owner name of the event:");

String ownerName = sc.nextLine();

System.out.println("Enter the type of the event:\n1.Exhibition\n2.StageEvent");

int choice = sc.nextInt();

switch(choice){

case 1 :

System.out.println("Enter the number of stalls:");

int noOfStalls = sc.nextInt();

Exhibition exObj = new Exhibition(name,detail,ownerName,noOfStalls);

totalrevenue = exObj.projectedRevenue();

System.out.println("The projected revenue of the event is " + totalrevenue);

break;

case 2 :

System.out.println("Enter the number of shows:");

int noOfShows = sc.nextInt();

System.out.println("Enter the number of seats per show:");

int noOfSeatsPerShow = sc.nextInt();

StageEvent seObj = new StageEvent(name,detail,ownerName,noOfShows,noOfSeatsPerShow);

totalrevenue = seObj.projectedRevenue();

System.out.println("The projected revenue of the event is " + totalrevenue);

break;

default : break;

}

}

}

public class Exhibition extends Event{

//Your code here

private Integer noOfStalls;

public Exhibition()

{

}

public Exhibition(String name, String detail, String ownerName,Integer noOfStalls)

{

super(name,detail,ownerName);

this.name = name;

this.detail = detail;

this.ownerName = ownerName;

this.noOfStalls = noOfStalls;

}

@Override

public double projectedRevenue()

{ double revenue = noOfStalls \* 10000;

return revenue;

}

}

**Java Interfaces**

Consider a Banking Scenario, There are many accounts, like Savings Account, Current Account, Demat Account and so on. We have a base Class Account which contains all the basic properties and methods of an Account. We do have some Maintainance Charges that apply to only some of the accounts. If you would like to enforce that the Savings Account & Current Account should have maintenance charges, then the simplest way is to ask your class implement the interface. If you do not implement the method in the class, it would raise a compilation error.

So, Java Interfaces essentially gives acts like a contract where its given that the methods declared in the interface has to be implemented in the class. Lets code the above Scenario.

Create a class named **Account** with following private attributes

|  |  |
| --- | --- |
| **Attribute** | **Datatype** |
| name | String |
| accountNumber | String |
| balance | double |
| startDate | String |

Include appropriate getters and setters.

Include appropriate parameterized constructors

Create an interface named **MaintenanceCharge** with the following method

|  |  |
| --- | --- |
| **Method** | **Description** |
| float calculateMaintancecharge(float no of years) | This method is used to calculate the maintenance charge |

Create a class named **CurrentAccount** which implements **MaintenanceCharge** interface  
Create a class named **SavingsAccount** which implements **MaintenanceCharge** interface

Create an another class named **Main** with main( ) method to test the above classes.

In Savings Account the maintenance amount will be 2\*m\*n+50.

In Current Account, the maintenance amount will be m\*n+200.

where m is the maintenance charge per year and n is the number of years.

Maintenance charge Rs.50  for saving account and 100 for the Current account.

**Note: Refere sample input and output for the specifications.**

**All text in bold corresponds to the input and remaining text corresponds to output.**

**Sample input and output 1:**

1.Current Account  
2.Savings Account  
**1**  
Name  
**SB**  
Account Number  
**12345**  
Account Balance  
**5000**  
Enter the Start Date(yyyy-mm-dd)  
**2013-04-22**  
Enter the Years  
**2**  
Maintenance Charge For Current Account 400.00

**Sample input and output 2:**  
1.Current Account  
2.Savings Account  
**2**  
Name  
**SB**  
Account Number  
**54321**  
Account Balance  
**3000**  
Enter the Start Date(yyyy-mm-dd)  
**2014-04-12**  
Enter the Years  
**5**  
Maintenance Charge For Savings Account 550.00

public class CurrentAccount implements MaintainanceCharge {

@Override

public float calculateMaintanceCharge(int noOfYear) {

return (100 \* noOfYear) + 200;

}

}

class Account {

String name;

String accountNumber;

double balance;

String startDate;

public Account(String name, String accountNumber, double balance, String startDate) {

super();

this.name = name;

this.accountNumber = accountNumber;

this.balance = balance;

this.startDate = startDate;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAccountNumber() {

return accountNumber;

}

public void setAccountNumber(String accountNumber) {

this.accountNumber = accountNumber;

}

public double getBalance() {

return balance;

}

public void setBalance(double balance) {

this.balance = balance;

}

public String getStartDate() {

return startDate;

}

public void setStartDate(String startDate) {

this.startDate = startDate;

}

}

public interface MaintainanceCharge{

float calculateMaintanceCharge(int noOfYear);

}

class SavingsAccount implements MaintainanceCharge {

@Override

public float calculateMaintanceCharge(int noOfYear) {

return (2 \* 50 \* noOfYear) + 50;

}

}

import java.text.DecimalFormat;

import java.util.\*;

public class Main {

public static void main(String args[]) {

Scanner s = new Scanner(System.in);

String name, sdate, account;

double accountBalance;

int years;

System.out.println("1.Current Account");

System.out.println("2.Savings Account");

int n = s.nextInt();

DecimalFormat df = new DecimalFormat("0.00");

switch (n) {

case 1:

System.out.println("Name");

name = s.next();

System.out.println("Account Number");

account = s.next();

System.out.println("Account Balance");

accountBalance = s.nextInt();

s.nextLine();

System.out.println("Enter the Start Date(yyyy-mm-dd)");

sdate = s.nextLine();

System.out.println("Enter the Years");

years = s.nextInt();

Account a = new Account(name, account, accountBalance, sdate);

CurrentAccount c = new CurrentAccount();

System.out

.println("Maintenance Charge For Current Account " + df.format(c.calculateMaintanceCharge(years)));

break;

case 2:

System.out.println("Name");

name = s.next();

System.out.println("Account Number");

account = s.next();

System.out.println("Account Balance");

accountBalance = s.nextInt();

s.nextLine();

System.out.println("Enter the Start Date(yyyy-mm-dd)");

sdate = s.nextLine();

System.out.println("Enter the Years");

years = s.nextInt();

Account a1 = new Account(name, account, accountBalance, sdate);

SavingsAccount sa = new SavingsAccount();

System.out

.println("Maintenance Charge For Savings Account " + df.format(sa.calculateMaintanceCharge(years)));

break;

default:

System.out.println("Invalid choice");

break;

}

}

}

**Interface**

The Interface defines a rule that any classes that implement it should override all the methods. Let's implement Interface in our application. We'll start simple, by including display method in the Stall interface. Now all types of stalls that implement the interface should override the method.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**  
  
Create an interface **Stall**with the following method

|  |  |
| --- | --- |
| **Method** | **Description** |
| void display() | abstract method. |

Create a class **GoldStall**which implements **Stall**interface with the following private attributes

|  |  |
| --- | --- |
| **Attribute** | **Datatype** |
| stallName | String |
| cost | Integer |
| ownerName | String |
| tvSet | Integer |

Create default constructor and a parameterized constructor with arguments in order **GoldStall(String stallName, Integer cost, String ownerName, Integer tvSet)**.  
Include appropriate getters and setters.  
  
Include the following method in the class **GoldStall**

|  |  |
| --- | --- |
| **Method** | **Description** |
| void display() | To display the stall name, cost of the stall, owner name and the number of tv sets. |

Create a class **PremiumStall**which implements **Stall**interface with following private attributes

|  |  |
| --- | --- |
| **Attribute** | **Datatype** |
| stallName | String |
| cost | Integer |
| ownerName | String |
| projector | Integer |

Create default constructor and a parameterized constructor with arguments in order **PremiumStall(String stallName, Integer cost, String ownerName, Integer projector)**.  
Include appropriate getters and setters.  
  
Include the following method in the class **PremiumStall.**

|  |  |
| --- | --- |
| **Method** | **Description** |
| void display() | To display the stall name, cost of the stall, owner name and the number of projectors. |

Create a class **ExecutiveStall**which implements **Stall**interface with following private attributes

|  |  |
| --- | --- |
| **Attribute** | **Datatype** |
| stallName | String |
| cost | Integer |
| ownerName | String |
| screen | Integer |

Create default constructor and a parameterized constructor with arguments in order **ExecutiveStall(String stallName, Integer cost, String ownerName, Integer screen)**.  
Include appropriate getters and setters.  
  
Include the following method in the class **ExecutiveStall.**

|  |  |
| --- | --- |
| **Method** | **Description** |
| void display() | To display the stall name, cost of the stall, owner name and the number of screens. |

Create a driver class named **Main** to test the above class.  
  
**Input Format:**  
The first input corresponds to choose the stall type.  
The next line of input corresponds to the details of the stall in CSV format according to the stall type.  
  
**Output Format:**  
Print “**Invalid Stall Type**” if the user has chosen the stall type other than the given type  
Otherwise, display the details of the stall.  
Refer to sample output for formatting specifications.  
  
**Note:** **All Texts in bold corresponds to the input and rest are output**  
  
**Sample Input and Output 1:**  
  
Choose Stall Type  
1)Gold Stall  
2)Premium Stall  
3)Executive Stall  
**1**  
Enter Stall details in comma separated(Stall Name,Stall Cost,Owner Name,Number of TV sets)  
**The Mechanic,120000,Johnson,10**  
Stall Name:The Mechanic  
Cost:120000.Rs  
Owner Name:Johnson  
Number of TV sets:10  
  
**Sample Input and Output 2:**  
  
ChooseStall Type  
1)Gold Stall  
2)Premium Stall  
3)Executive Stall  
**2**  
Enter Stall details in comma separated(Stall Name,Stall Cost,Owner Name,Number of Projectors)  
**Knitting plaza,300000,Zain,20**  
Stall Name:Knitting plaza  
Cost:300000.Rs  
Owner Name:Zain  
Number of Projectors:20  
  
**Sample Input Output 3:**  
  
ChooseStall Type  
1)Gold Stall  
2)Premium Stall  
3)Executive Stall  
**3**  
Enter Stall details in comma separated(Stall Name,Stall Cost,Owner Name,Number of Screens)  
**Fruits Hunt,10000,Uber,7**  
Stall Name:Fruits Hunt  
Cost:10000.Rs  
Owner Name:Uber  
Number of Screens:7  
  
**Sample Input Output 4:**  
  
ChooseStall Type  
1)Gold Stall  
2)Premium Stall  
3)Executive Stall  
**4**  
Invalid Stall Type

public class PremiumStall implements Stall {

private String stallName;

private Integer cost;

private String ownerName;

private Integer projector;

public PremiumStall(String stallName, Integer cost, String ownerName, Integer projector) {

this.stallName = stallName;

this.cost = cost;

this.ownerName = ownerName;

this.projector = projector;

}

public PremiumStall() {

}

public void display() {

System.out.println("Stall Name:" + stallName + "\nCost:" + cost + ".Rs\nOwner Name:" + ownerName

+ "\nNumber of Projectors:" + projector);

}

public String getStallName() {

return stallName;

}

public void setStallName(String stallName) {

this.stallName = stallName;

}

public Integer getCost() {

return cost;

}

public void setCost(Integer cost) {

this.cost = cost;

}

public String getOwnerName() {

return ownerName;

}

public void setOwnerName(String ownerName) {

this.ownerName = ownerName;

}

public Integer getProjector() {

return projector;

}

public void setProjector(Integer projector) {

this.projector = projector;

}

}

public class ExecutiveStall implements Stall {

private String stallName;

private Integer cost;

private String ownerName;

private Integer screen;

public ExecutiveStall(String stallName, Integer cost, String ownerName, Integer screen) {

this.stallName = stallName;

this.cost = cost;

this.ownerName = ownerName;

this.screen = screen;

}

public ExecutiveStall() {

}

public void display() {

System.out.println("Stall Name:" + stallName + "\nCost:" + cost + ".Rs\nOwner Name:" + ownerName

+ "\nNumber of Screens:" + screen);

}

public String getStallName() {

return stallName;

}

public void setStallName(String stallName) {

this.stallName = stallName;

}

public Integer getCost() {

return cost;

}

public void setCost(Integer cost) {

this.cost = cost;

}

public String getOwnerName() {

return ownerName;

}

public void setOwnerName(String ownerName) {

this.ownerName = ownerName;

}

public Integer getScreen() {

return screen;

}

public void setScreen(Integer screen) {

this.screen = screen;

}

}

public interface Stall {

void display();

}

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws Exception {

// BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String[] arr = new String[3];

String stallDetails = "";

System.out.println("Choose Stall Type\n1)Gold Stall\n2)Premium Stall\n3)Executive Stall");

Scanner scan = new Scanner(System.in);

int n = scan.nextInt();

switch (n) {

case 1:

scan.nextLine();

System.out.println(

"Enter Stall details in comma separated(Stall Name,Stall Cost,Owner Name,Number of TV sets)");

stallDetails = scan.nextLine();

arr = stallDetails.split(",");

int cost = Integer.parseInt(arr[1]);

int num = Integer.parseInt(arr[3]);

GoldStall gStall = new GoldStall(arr[0], cost, arr[2], num);

gStall.display();

break;

case 2:

scan.nextLine();

System.out.println(

"Enter Stall details in comma separated(Stall Name,Stall Cost,Owner Name,Number of Projectors)");

stallDetails = scan.nextLine();

arr = stallDetails.split(",");

cost = Integer.parseInt(arr[1]);

num = Integer.parseInt(arr[3]);

PremiumStall pStall = new PremiumStall(arr[0], cost, arr[2], num);

pStall.display();

break;

case 3:

scan.nextLine();

System.out.println(

"Enter Stall details in comma separated(Stall Name,Stall Cost,Owner Name,Number of Screens)");

stallDetails = scan.nextLine();

arr = stallDetails.split(",");

cost = Integer.parseInt(arr[1]);

num = Integer.parseInt(arr[3]);

ExecutiveStall eStall = new ExecutiveStall(arr[0], cost, arr[2], num);

eStall.display();

break;

default:

System.out.println("Invalid Stall Type");

}

}

}

public class GoldStall implements Stall {

private String stallName;

private Integer cost;

private String ownerName;

private Integer tvSet;

public GoldStall(String stallName, Integer cost, String ownerName, Integer tvSet) {

this.stallName = stallName;

this.cost = cost;

this.ownerName = ownerName;

this.tvSet = tvSet;

}

public GoldStall() {

}

public void display() {

System.out.println("Stall Name:" + stallName + "\nCost:" + cost + ".Rs\nOwner Name:" + ownerName

+ "\nNumber of TV sets:" + tvSet);

}

public String getStallName() {

return stallName;

}

public void setStallName(String stallName) {

this.stallName = stallName;

}

public Integer getCost() {

return cost;

}

public void setCost(Integer cost) {

this.cost = cost;

}

public String getOwnerName() {

return ownerName;

}

public void setOwnerName(String ownerName) {

this.ownerName = ownerName;

}

public Integer getTvSet() {

return tvSet;

}

public void setTvSet(Integer tvSet) {

this.tvSet = tvSet;

}

}

**Round up - Interfaces**

In one of the earlier exercises in Interfaces, we looked at how banks encrypt & decrypt transaction details to ensure safety. We also saw understood that each of them complies with the methods specified by the Governing Agency in the form of an interface. To Round-off interfaces, let's look at a similar example, where the governing agency mandates that any customer who performs a transaction has to be notified through SMS, Email and a monthly e-statement. As expected, we define an interface Notification and three methods as specified below. Lets code this example.

Create an interface named **Notification** with the following methods

    notificationBySms( ),  
    notificationByEmail( ),  
    notificationByCourier( ).

Create a class named **ICICI** which implements **Notification**interface

Create a class named **HDFC** which implements **Notification**interface  
  
Create a class **BankFactory**with two methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| public Icici getIcici( ) | This method is used to return the object for ICICI class |
| public Hdfc getHdfc( ) | This method is used to return the object for HDFC class |

Create the **Main**class with main( ) method to test the above class.  
  
**Input and Output format:**  
The first integer corresponds to select the bank, the next integer corresponds to the type of the notification.  
If there is no valid input then display '**Invalid input**'.

**[Note: All text in bold corresponds to the input and remaining text corresponds to output]**  
  
**Sample Input and Output 1:**  
  
Welcome to Notification Setup  
Please select your bank:  
1)ICICI  
2)HDFC  
**1**  
Enter the type of Notification you want to enter  
1)SMS  
2)Mail  
3)Courier  
**1**  
ICICI - Notification By SMS  
  
**Sample Input and Output 2:**  
  
Welcome to Notification Setup  
Please select your bank:  
1)ICICI  
2)HDFC  
**2**  
Enter the type of Notification you want to enter  
1)SMS  
2)Mail  
3)Courier  
**3**  
HDFC - Notification By Courier  
  
**Sample Input and Output 3:**  
  
Welcome to Notification Setup  
Please select your bank:  
1)ICICI  
2)HDFC  
**3**  
Invalid Input

public class ICICI implements Notification {

@Override

public void notificationBySms() {

System.out.println("ICICI - Notification By SMS");

}

@Override

public void notificationByEmail() {

System.out.println("ICICI - Notification By Mail");

}

@Override

public void notificationByCourier() {

System.out.println("ICICI - Notification By Courier");

}

}

public class HDFC implements Notification {

@Override

public void notificationBySms() {

System.out.println("HDFC - Notification By SMS");

}

@Override

public void notificationByEmail() {

System.out.println("HDFC - Notification By Mail");

}

@Override

public void notificationByCourier() {

System.out.println("HDFC - Notification By Courier");

}

}

public interface Notification {

public void notificationBySms();

public void notificationByEmail();

public void notificationByCourier();

}

public class BankFactory {

public ICICI getIcici() {

ICICI icici = new ICICI();

return icici;

}

public HDFC getHdfc() {

HDFC hdfc = new HDFC();

return hdfc;

}

}

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws IOException {

Scanner sc = new Scanner(System.in);

System.out.println("Welcome to Notification Setup");

System.out.println("Please select your bank:\n1)ICICI\n2)HDFC");

int bank = sc.nextInt();

switch (bank) {

case 1:

System.out.println("Enter the type of Notification you want to enter\n1)SMS\n2)Mail\n3)Courier");

int msg = sc.nextInt();

BankFactory b = new BankFactory();

ICICI c = b.getIcici();

switch (msg) {

case 1:

c.notificationBySms();

break;

case 2:

c.notificationByEmail();

break;

case 3:

c.notificationByCourier();

break;

default:

System.out.println("Invalid Input");

break;

}

break;

case 2:

System.out.println("Enter the type of Notification you want to enter\n1)SMS\n2)Mail\n3)Courier");

int msg1 = sc.nextInt();

BankFactory b1 = new BankFactory();

HDFC h = b1.getHdfc();

switch (msg1) {

case 1:

h.notificationBySms();

break;

case 2:

h.notificationByEmail();

break;

case 3:

h.notificationByCourier();

break;

default:

System.out.println("Invalid Input");

break;

}

break;

default:

System.out.println("Invalid Input");

break;

}

}

}

**Static Inner Class**

Write a program to calculate the area of the rectangle and triangle using the static inner class concept in java.

Create an outer class **Shape** with the following **public static**attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| value1 | Double |
| value2 | Double |

Create a static inner class **Rectangle** which have the outer class **Shape.**  
Include the following method in the **Rectangle** class

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public Double computeRectangleArea() | Here Calculate and return the area of the rectangle by accessing the attributes value1(length) & value2(breadth) of Shape class. Area of the rectangle = (length \* breadth) |

Create a static inner class **Triangle** which have the outer class **Shape.**  
Include the following method in the **Triangle** class

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public Double computeTriangleArea() | Here Calculate and return the area of the triangle by accessing the attributes value1(base) & value2(height) of Shape class. Area of the triangle = (1/2) \* (base \* height) |

Get the option for the shape to compute the area and get the attribute according to the shape option and set the values to the Shape class attributes. Calculate the area and print the area.  
While printing round off the area to 2 decimal formats.  
  
Create a driver class **Main** to test the above classes.  
  
**[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]**  
  
**Input Format**  
The first line of the input is an integer corresponds to the shape.  
The next line of inputs are Double which corresponds to,  
For Rectangle(Option 1) get the length and breadth.  
For Triangle(Option 2) get the base and height.  
  
**Output Format**  
The output consists area of the shape.  
Print the double value correct to two decimal places.  
Print “**Invalid choice**”, if the option for the shape is chosen other than the given options.  
Refer to sample output for formatting specifications.  
  
**[All text in bold corresponds to input and rest corresponds to output]  
Sample Input/Output 1:**  
Enter the shape  
1.Rectangle  
2.Triangle  
**1**  
Enter the length and breadth:  
**10  
25**  
Area of rectangle is 250.00  
  
**Sample Input/Output 2:**  
Enter the shape  
1.Rectangle  
2.Triangle  
**2**  
Enter the base and height:  
**15  
19**  
Area of triangle is 142.50  
  
**Sample Input/Output 3:**  
Enter the shape  
1.Rectangle  
2.Triangle  
**3**  
Invalid choice

Top of Form

Bottom of Form

public class Shape {

public static Double value1;

public static Double value2;

public static class Rectangle {

public Double computeRectangleArea() {

return (value1 \* value2);

}

}

public static class Triangle {

public Double computeTriangleArea() {

return ((0.5) \* (value1 \* value2));

}

}

}

import java.io.IOException;

import java.text.DecimalFormat;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws IOException {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the shape\r\n1.Rectangle\r\n2.Triangle");

int ch = sc.nextInt();

DecimalFormat df = new DecimalFormat("0.00");

switch (ch) {

case 1:

System.out.println("Enter the length and breadth:");

Shape.value1 = sc.nextDouble();

Shape.value2 = sc.nextDouble();

Shape.Rectangle r = new Shape.Rectangle();

Double rectangle = r.computeRectangleArea();

System.out.println("Area of rectangle is " + df.format(rectangle));

break;

case 2:

System.out.println("Enter the base and height:");

Shape.value1 = sc.nextDouble();

Shape.value2 = sc.nextDouble();

Shape.Triangle t = new Shape.Triangle();

Double triangle = t.computeTriangleArea();

System.out.println("Area of triangle is " + df.format(triangle));

break;

default:

System.out.println("Invalid choice");

break;

}

}

}

**Article count**

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such a program is called a thread. The threads are light-weight processes within a process.  
           
Let's have a quick look at the way threads work in Java. For multi-threading to work, the class that will be invoked as a thread should extend the Thread class. You may wonder, what is the use of multi-threading. Let's understand it by the following exercise. Given 'n' number of lines of text, you have to find the total number of articles present in the given lines. while obtaining inputs from the user, the Main method has the full control of the execution.  
  
The time is wasted in input gathering, which can be invaluable for large computing applications, has to be utilized properly. Hence a thread is invoked when a line is obtained and the articles are counted while the input for the subsequent lines is obtained from the user. Thus threading can increase efficiency and time constraints.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**  
   
Create a class called **Article**which extends the **Thread**class with the following private attributes.

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| line | String |
| count | Integer |

Include appropriate getters and setters.  
Generate default and parameterized constructors. The format for the parameterized constructor is **Article(String line)**  
  
The Article class includes the following methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| void run() | This method counts the number of articles in a given line and stores the value in the count variable. |

Create a driver class called **Main**. In the Main method, invoke 'n' threads for 'n' lines of input and compute the total count of the articles in the given lines.  
  
**Input and Output format:**  
Refer to sample Input and Output for formatting specifications.

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

**Sample Input and Output:**

Enter the number of lines

**3**

Enter line 1

**An article is a word used to modify a noun, which is a person, place, object, or idea.**

Enter line 2

**Technically, an article is an adjective, which is any word that modifies a noun.**

Enter line 3

**There are two different types of articles.**

There are 7 articles in the given input

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class Article extends Thread {

String line;

int count;

public Article() {

super();

}

public Article(String line) {

super();

this.line = line;

}

public String getLine() {

return line;

}

public void setLine(String line) {

this.line = line;

}

public int getCount() {

return count;

}

public void setCount(int count) {

this.count = count;

}

@Override

public void run() {

Matcher m = Pattern.compile("(?i)\\b((a)|(an)|(the))\\b").matcher(this.getLine());

while (m.find()) {

this.count++;

}

}

}

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws InterruptedException{

Scanner sc = new Scanner(System.in);

System.out.println("Enter the number of lines");

int n = sc.nextInt();

int count = 0;

sc.nextLine();

for(int i=0;i<n;i++) {

Article a = new Article();

System.out.println("Enter line "+(i+1));

a.setLine(sc.nextLine());

a.start();

a.join();

count +=a.getCount();

}

System.out.println("There are "+count+" articles in the given input");

}

}

**Profit or Loss**

We are going to create a console application that can estimate whether the booking is a profit or loss, thereby enabling hall owners to reduce or increase expenses depending on the status. Hence if several Booking details are given, compute whether the bookings are profitable or not. Use Threads to compute for each booking, Finally display the details along with the profit/loss status.  
  
**Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.**

Create a class **Event**with the following private attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| name | String |
| hallbooking | HallBooking |

Include appropriate getters and setters.  
Create default constructor and a parameterized constructor with arguments in order**Event(String name,HallBooking hallbooking).**

Create a class **HallBooking**with following private attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| hallName | String |
| cost | Double |
| hallCapacity | Integer |
| seatsBooked | Integer |

Include appropriate getters and setters.  
Create default constructor and a parameterized constructor with arguments in order  
**HallBooking(String hallName, Double cost, Integer hallCapacity,Integer seatsBooked)**.

Create a class **ComputeStatus** that implements **Runnable interface** with **List<Event> eventList** attribute.  
  
Include following methods.  
**Override run() method** which displays event name along with their status (i.e) Profit or Loss.   
**If  (seats booked / hall capacity) \* 100 >= 60 then it is a profit else loss**.  
  
Create a driver class **Main** which creates **ThreadGroup** with two threads. Each thread will have half of the event details. After the first half of event details are obtained, invoke the first thread and after the other half is obtained invoke the second thread. The Threads print the status of the events. Use the join method appropriately to ensure printing the status in the correct order.

**Input Format:**  
  
The first line of input corresponds to the number of events 'n'.  
The next 'n' line of input corresponds to the event details in CSV format of (Event Name,Hall Name,Cost,Hall Capacity,Seats Booked).  
Refer to sample input for formatting specifications.  
  
**Output Format:**  
  
The output consists of event names with their status (Profit or Loss).  
Refer to sample output for formatting specifications.  
  
**Problem Constraints:**  
If n>0 and n then even. Otherwise, display as "**Invalid Input**".  
  
**Sample Input and Output 1:**  
**[All text in bold corresponds to input and rest corresponds to output]**  
  
Enter the number of events  
**4**  
Enter event details in CSV  
**Party,Le Meridian,12000,400,250  
Wedding,MS mahal,500000,1000,400  
Alumni meet,Ramans,10000,600,300  
Plaza party,Rizzodous,30000,1200,1000**  
Party yields profit  
Wedding yields loss  
Alumni meet yields loss  
Plaza party yields profit

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import java.util.Scanner;

public class Event {

private String name;

private HallBooking hallBooking;

public Event() {

}

public Event(String name, HallBooking hallBooking) {

this.name = name;

this.hallBooking = hallBooking;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public HallBooking getHallBooking() {

return hallBooking;

}

public void setHallBooking(HallBooking hallBooking) {

this.hallBooking = hallBooking;

}

}

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import java.util.Scanner;

public class ComputeStatus implements Runnable {

List<Event> eventList;

public List<Event> getEventList() {

return eventList;

}

public void setEventList(List<Event> eventList) {

this.eventList = eventList;

}

@Override

public void run() {

// System.out.println(Thread.currentThread().getName());

Iterator iterator = eventList.iterator();

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

while (iterator.hasNext()) {

Event e = (Event) iterator.next();

if (e.getHallBooking().getSeatsBooked() \* 100 / e.getHallBooking().getHallCapacity() >= 60) {

System.out.println(e.getName() + " yields profit");

} else {

System.out.println(e.getName() + " yields loss");

}

}

}

}

public class HallBooking {

private String hallName;

private Double cost;

private Integer hallCapacity;

private Integer seatsBooked;

public HallBooking(String hallName, Double cost, Integer hallCapacity, Integer seatsBooked) {

this.hallName = hallName;

this.cost = cost;

this.hallCapacity = hallCapacity;

this.seatsBooked = seatsBooked;

}

public HallBooking() {

}

public String getHallName() {

return hallName;

}

public void setHallName(String hallName) {

this.hallName = hallName;

}

public Double getCose() {

return cost;

}

public void setCose(Double cose) {

this.cost = cose;

}

public Integer getHallCapacity() {

return hallCapacity;

}

public void setHallCapacity(Integer hallCapacity) {

this.hallCapacity = hallCapacity;

}

public Integer getSeatsBooked() {

return seatsBooked;

}

public void setSeatsBooked(Integer seatsBooked) {

this.seatsBooked = seatsBooked;

}

}

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the number of events");

int noOfEvents = Integer.parseInt(scanner.nextLine());

if (noOfEvents > 0 && noOfEvents % 2 == 0) {

System.out.println("Enter event details in CSV");

String eventDetail = "";

List<Event> eventList = new ArrayList<Event>();

ComputeStatus computeStatus = new ComputeStatus();

// computeStatus.setEventList(eventList);

ThreadGroup threadGroup = new ThreadGroup("TG1");

Thread t1 = new Thread(threadGroup, computeStatus, "T1");

Thread t2 = new Thread(threadGroup, computeStatus, "T2");

for (int i = 0; i < noOfEvents; i++) {

eventDetail = scanner.nextLine();

String[] details = eventDetail.split(",");

HallBooking hallBooking = new HallBooking(details[1],

Double.parseDouble(details[2]),

Integer.parseInt(details[3]),

Integer.parseInt(details[4]));

Event event = new Event(details[0], hallBooking);

eventList.add(event);

computeStatus.setEventList(eventList);

if (i + 1 == noOfEvents / 2) {

t1.start();

try {

t1.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

eventList.clear();

}

if (i + 1 == noOfEvents) {

t2.start();

}

}

} else {

System.out.println("Invalid Input");

}

}

}

**Multi-Threading**

To illustrate the creation of multiple threads in a program performing concurrent operations, let us consider the processing of the following mathematical equation:  
**p = sin (x) + cos (y) + tan (z)**  
As these trigonometric functions are independent operations without any dependencies between them, they can be executed concurrently. After that their results can be combined to produce the final result.  
  
All three worker threads are concurrently executed on shared or dedicated CPUs depending on the type of machine. Although the master thread can continue its execution, in this case, it needs to make sure that all operations are completed before combining individual results. This is accomplished by waiting for each thread to complete by invoking **join()** method associated with each worker thread.  
  
The main thread is called **Main**, which acts like a master thread. It creates three worker threads (**SineClass**, **CosClass**, and **TanClass**) and assigns them to compute values for different data inputs.  
  
**Input & Output Format:**  
Refer sample Input and Output for formatting specifications.  
  
**Hint:**  
Use the following code snippet to print to 2 decimal places.  
import java.text.DecimalFormat;  
DecimalFormat df = new DecimalFormat("#.##");  
System.out.println("Sum of sin, cos, tan = " + df.format(z));  
  
**Sample Input and Output :**  
Enter the Degree for Sin :  
**45**  
Enter the Degree for Cos :  
**30**  
Enter the Degree for Tan :  
**30**  
Sum of sin, cos, tan = 2.15

public class CosClass extends Thread {

double value;

double cos;

public CosClass(double cos) {

// TODO Auto-generated constructor stub

this.cos = cos;

}

@Override

public void run() {

double b = Math.toRadians(cos);

value = Math.cos(b);

}

public double getValue() {

return value;

}

}

public class SineClass extends Thread {

double value;

double sins;

public SineClass(double sins) {

this.sins = sins;

}

@Override

public void run() {

double b = Math.toRadians(sins);

value = Math.sin(b);

}

public double getValue() {

return value;

}

}

public class TanClass extends Thread {

double value;

double tan;

public TanClass(double tan) {

// TODO Auto-generated constructor stub

this.tan = tan;

}

@Override

public void run() {

double b = Math.toRadians(tan);

value = Math.tan(b);

}

public double getValue() {

return value;

}

}

import java.text.DecimalFormat;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

// TODO Auto-generated method stub

DecimalFormat df = new DecimalFormat("0.00");

Scanner sc = new Scanner(System.in);

System.out.println("Enter the Degree for Sin : ");

double sins = sc.nextDouble();

System.out.println("Enter the Degree for Cos : ");

double coss = sc.nextDouble();

System.out.println("Enter the Degree for Tan : ");

double tans = sc.nextDouble();

SineClass s = new SineClass(sins);

CosClass c = new CosClass(coss);

TanClass t = new TanClass(tans);

s.start();

try {

s.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

c.start();

try {

c.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

t.start();

try {

t.join();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("Sum of sin, cos, tan = " + df.format(s.getValue() + t.getValue() + c.getValue()));

}

}Top of Form

Bottom of Form

**Wrapper Class – 1**

Write a Java program to print the following static values defined in the Float Wrapper Class  
  
Maximum exponent a float can hold  
Maximum value a float can hold  
Number of bits used to represent a float value  
  
**Input and Output Format:**  
  
There is no input to this program.  
Refer sample output for formatting specifications.  
  
**Sample Output:**  
Maximum exponent :127  
Maximum value :3.4028235E38  
Number of bits :32

Top of Form

Bottom of Form

public class Main {

public static void main(String[] args) {

System.out.println("Maximum exponent :"+Float.MAX\_EXPONENT+" ");

System.out.println("Maximum value :"+Float.MAX\_VALUE+" ");

System.out.println("Number of bits :"+Float.SIZE );

}

}

**Converting a String to Double**

Many a times we receive the input in String and convert it into other data types (Double / Integer). Lets practice this simple exercise. Get the input amount as string and parse it using 'valueOf/parseDouble' method.

Create a main class "**Main.java**".

Create another class file "**BillHeader.java**" with the following private members

|  |  |
| --- | --- |
| **Data Type** | **Variable name** |
| Date | issueDate |
| Date | dueDate |
| Double | originalAmount |
| Double | amountOutstanding |

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

**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 the issue date as dd/MM/yyyy

**12/07/2015**

Enter the due date as dd/MM/yyyy

**21/08/2015**

Enter the original amount

**2000**

Enter amount paid so far

**1000**

Issue date: 12/07/2015

Due Date: 21/08/2015

Original amount Rs.2000.0

Amount outstanding Rs.1000.0

import java.util.Date;

import java.util.Scanner;

import java.io.IOException;

import java.text.\*;

public class Main {

public static void main(String[] args) throws IOException, ParseException {

Scanner sc = new Scanner(System.in);

SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");

System.out.println("Enter the issue date as dd/MM/yyyy");

Date issueDate = formatter.parse(sc.nextLine());

System.out.println("Enter the due date as dd/MM/yyyy");

Date dueDate = formatter.parse(sc.nextLine());

System.out.println("Enter the original amount");

Double originalAmount = Double.parseDouble(sc.nextLine());

System.out.println("Enter amount paid so far");

Double paidAmount = Double.parseDouble(sc.nextLine());

Double amountOutstanding = originalAmount - paidAmount;

BillHeader billHeader = new BillHeader();

billHeader.setIssueDate(issueDate);

billHeader.setDueDate(dueDate);

billHeader.setOriginalAmount(originalAmount);

billHeader.setAmountOutstanding(amountOutstanding);

//System.out.println("Issue date: " + formatter.format(billHeader.getIssueDate()) + "\nDue Date: " + formatter.format(billHeader.getDueDate()) + "\nOriginal amount Rs." + billHeader.getOriginalAmount() + "\nAmount outstanding Rs." + billHeader.getAmountOutstanding());

System.out.println(billHeader);

}

}

import java.util.Date;

import java.text.\*;

public class BillHeader {

private Date issueDate;

private Date dueDate;

private Double originalAmount;

private Double amountOutstanding;

public BillHeader() {

}

public BillHeader(Date issueDate, Date dueDate, Double originalAmount, Double amountOutstanding) {

super();

this.issueDate = issueDate;

this.dueDate = dueDate;

this.originalAmount = originalAmount;

this.amountOutstanding = amountOutstanding;

}

public Date getIssueDate() {

return issueDate;

}

public void setIssueDate(Date issueDate) {

this.issueDate = issueDate;

}

public Date getDueDate() {

return dueDate;

}

public void setDueDate(Date dueDate) {

this.dueDate = dueDate;

}

public Double getOriginalAmount() {

return originalAmount;

}

public void setOriginalAmount(Double originalAmount) {

this.originalAmount = originalAmount;

}

public Double getAmountOutstanding() {

return amountOutstanding;

}

public void setAmountOutstanding(Double amountOutstanding) {

this.amountOutstanding = amountOutstanding;

}

@Override

public String toString() {

SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");

//return "Issue date: " + issueDate + "\nDue Date: " + dueDate + "\nOriginal amount Rs." + originalAmount + "\nAmount outstanding Rs." + amountOutstanding;

return "Issue date: " + formatter.format(getIssueDate()) + "\nDue Date: " + formatter.format(getDueDate()) + "\nOriginal amount Rs." + getOriginalAmount() + "\nAmount outstanding Rs." + getAmountOutstanding();

}

}

**String Reversal**

Write a program to reverse a given string.

**Input and Output Format:**

Input consists of a string.

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 a string to reverse

**Punitha**

Reverse of entered string is : ahtinuP

import java.lang.\*;

import java.io.\*;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws IOException {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a string to reverse");

String str = sc.nextLine();

StringBuffer sbr = new StringBuffer(str);

// To reverse the string

System.out.println("Reverse of entered string is : " + sbr.reverse());

}

}

**String API : startsWith() : Illustration**

Write a program to illustrate the use of the method startsWith() defined in the string API.

**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:**

Enter the string

**Ampphisoft**

Enter the start string

**Amphi**

"Ampphisoft" does not start with "Amphi"

**Sample Input and Output 2:**

Enter the string

**Amphisoft**

Enter the start string

**Amphi**

"Amphisoft" starts with "Amphi"

import java.util.\*;

public class Main {

public static void main(String[] args) {

// fill your code here

Scanner scan = new Scanner(System.in);

System.out.println("Enter the string");

String str = scan.nextLine();

System.out.println("Enter the start string");

// fill your code here

String chk = scan.nextLine();

if(str.startsWith(chk) )

{

System.out.println('"' + str + '"'+ " starts with "+ '"'+ chk+'"');

}

else

{

System.out.println('"' + str + '"'+ " does not start with "+ '"'+ chk+'"');

}

}

}

**String API : split() : Illustration**

This program is to illustrate the use of the method **split()** defined in the string API.

Write a program to split a string based on spaces (There may be multiple spaces too) and returns the tokens in the form of an array.

**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 string

**Amphisoft Technologies is             a                  private     organization**

The words in the string are

Amphisoft

Technologies

is

a

private

organization

import java.util.\*;

public class Main {

public static void main(String[] args) {

// fill your code here

Scanner scan = new Scanner(System.in);

System.out.println("Enter the string");

String str = scan.nextLine();

String str2 = str.replaceAll(" +", " ");

String str1[] = str2.split(" ",-1);

System.out.println("The words in the string are");

for(int i = 0; i < str1.length; i++)

{

System.out.println(str1[i]);

}

}

}

**String Tokenizer**

Write a Java program to implement string tokenizer inorder to split a string into two different tokens by =(equal to) and ;(semicolon).  
  
**Input Format:**  
Input is a string which needs to be split.  
  
**Output Format:**  
Each line of the Output contains two strings. The first string is formed by token '=' and the second string is formed by the token ';'  
  
**Assume**: The tokens, '=' and ';', will always come alternately. Refer Sample Input.

**Sample Input:**

title=Java-Samples;author=Emiley J;publisher=java-samples.com;copyright=2007;

**Sample Output:**

title Java-Samples

author Emiley J

publisher java-samples.com

copyright 2007

Top of Form

Bottom of Form

import java.util.Scanner;

import java.util.StringTokenizer;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String s = sc.nextLine();

StringTokenizer s1 = new StringTokenizer(s, "=");

String s2 = "";

while(s1.hasMoreTokens()) {

s2 = s2+" "+s1.nextToken();

}

StringTokenizer s3 = new StringTokenizer(s2, ";");

while(s3.hasMoreTokens()) {

System.out.println(s3.nextToken());

}

}

}

**Customer Address Using String Builder**

We know that Strings are immutable and are placed in a common pool called String Pool. It is always suggested that if the value of a string variable changes quite often, then the string pool would keep creating new strings and is considered as a bad practice. In these cases, the alternate way is to use StringBuilder & StringBuffer.  
  
StringBuilder would append values in a normal heap instead of any common string pools.  
The only difference between StringBuilder & StringBuffer is: StringBuilder is not Thread-Safe whereas StringBuffer is Thread-Safe.  
Let's try out using StringBuilder.  
  
Write a Java Program to display the address of the customer in a particular format.

Create a main class "**Main.java**".

Create another class file "**Address.java**" with the following private members.

|  |  |
| --- | --- |
| **Data Type** | **Variable name** |
| String | line1 |
| String | line2 |
| String | city |
| String | country |
| int | zipCode |

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

Use '**toString**' method in the Address class and append using String Builder.

**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 Address Details :

Enter Line 1 :

**152, South Block**

Enter Line 2 :

**Raisina Hill**

Enter City :

**New Delhi**

Enter Country :

**India**

Enter Zip Code :

**110011**

Address Details :

152, South Block,

Raisina Hill,

New Delhi - 110011

India

// fill your code here

public class Address {

private String line1;

private String line2;

private String city;

private String country;

private int zipCode;

public Address() {

super();

// TODO Auto-generated constructor stub

}

public Address(String line1, String line2, String city, String country, int zipCode) {

super();

this.line1 = line1;

this.line2 = line2;

this.city = city;

this.country = country;

this.zipCode = zipCode;

}

public String getLine1() {

return line1;

}

public void setLine1(String line1) {

this.line1 = line1;

}

public String getLine2() {

return line2;

}

public void setLine2(String line2) {

this.line2 = line2;

}

public String getCity() {

return city;

}

public void setCity(String city) {

this.city = city;

}

public String getCountry() {

return country;

}

public void setCountry(String country) {

this.country = country;

}

public int getZipCode() {

return zipCode;

}

public void setZipCode(int zipCode) {

this.zipCode = zipCode;

}

@Override

public String toString() {

return "Address Details :\n" + new StringBuilder().append(this.getLine1()) + ",\n" + new StringBuilder().append(this.getLine2()) + ",\n"

+ new StringBuilder().append(this.getCity()) + " - " + new StringBuilder().append(this.getZipCode()) + "\n"

+ new StringBuilder().append(this.getCountry());

}

}

import java.io.IOException;

import java.util.Scanner;

public class Main {

public static void main(String[] args) throws IOException {

// fill your code here

Scanner sc = new Scanner(System.in);

System.out.println("Enter Address Details :\nEnter Line 1 :");

String line1 = sc.nextLine();

System.out.println("Enter Line 2 :");

String line2 = sc.nextLine();

System.out.println("Enter City :");

String city = sc.nextLine();

System.out.println("Enter Country :");

String country = sc.nextLine();

System.out.println("Enter Zip Code :");

int zipCode = sc.nextInt();

Address address = new Address(line1, line2, city, country, zipCode);

System.out.println(address.toString());

}

}

**String API : endsWith() : Illustration**

Write a program to illustrate the use of the method **endsWith()** defined in the string API.

**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:**

Enter the string

**Ampphisoft**

Enter the end string

**softi**

"Ampphisoft" does not end with "softi"

**Sample Input and Output 2:**

Enter the string

**Amphisoft**

Enter the end string

**soft**

"Amphisoft" ends with "soft"

Top of Form

Bottom of Form

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scnr = new Scanner(System.in);

System.out.println("Enter the string");

String str = scnr.nextLine();

System.out.println("Enter the end string");

String strEnd = scnr.nextLine();

if( str.endsWith( strEnd ) ) {

System.out.println( "\"" + str + "\"" + " ends with \"" + strEnd + "\"" );

} else {

System.out.println( "\"" + str + "\"" + " does not end with \"" + strEnd + "\"" );

}

}

}

**Wrapper Class – Integer II**

This program is to illustrate the parseInt() method defined in the Integer Wrapper class.

Write a program that accepts 3 String values as input and invokes some of the methods defined in the Integer Wrapper class.

Refer sample input and output. All functions should be performed using the methods defined in the Integer class.

**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 binary number

**111**

Enter the octal number

**11**

Enter the hexadecimal number

**1F**

The integer value of the binary number 111 is 7

The integer value of the octal number 11 is 9

The integer value of the hexadecimal number 1F is 31

Top of Form

Bottom of Form

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scnr = new Scanner(System.in);

System.out.println("Enter the binary number");

String binaryNo = scnr.next();

System.out.println("Enter the octal number");

String octalNo = scnr.next();

System.out.println("Enter the hexadecimal number");

String hexaDecimalNo = scnr.next();

int binNo = Integer.parseInt(binaryNo,2);

int octNo = Integer.parseInt(octalNo,8);

int hexNo = Integer.parseInt(hexaDecimalNo,16);

System.out.println( "The integer value of the binary number " + binaryNo + " is " + binNo );

System.out.println( "The integer value of the octal number " + octalNo + " is "+ octNo );

System.out.println( "The integer value of the hexadecimal number " + hexaDecimalNo + " is "+ hexNo);

}

}

**String API : replace() : Illustration**

Write a program to illustrate the use of the method **replace()** defined in the string API.

Two companies enter into a Marketing Agreement and they prepare an Agreement Draft. After that one of the companies changes its name. The name changes need to be made in the Agreement Draft as well. Write a program to perform the name changes in the agreement draft.

**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 content of the document

**Amphi is a private organisation. Amphi is a product based company. EBox is a Amphi product**

Enter the old name of the company

**Amphi**

Enter the new name of the company

**Amphisoft**

The content of the modified document is

Amphisoft is a private organisation. Amphisoft is a product based company. EBox is a Amphisoft product

Top of Form

Bottom of Form

**Equals & == in String**

One important property of String literals are that - All String literals are placed in a common pool called String pool. If I create two string literals with the same content, both will point to the same object. Moreover, If i reassign the variable to another value, a new space is created and the existing space is left with the same value. Once a value is assigned or allocated it is never modified until its destroyed. This is why strings are called immutable.  
"==" operator compares address, but since string literals point to the same location (if values are same) it returns true.  
"equals" operator compares the content of the two strings.  
  
Lets check the above concept by validating Emailids of two customers first by using equals & equalsIgnoreCase methods.

Create a main class "**Main**.**java**".

Create another class file "**Customer**.**java**" with following private member variables.

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

Include appropriate **getters** and **setters**.  
  
Use '**equals()**' and '**equalsIgnoreCase()**' to compare the email ids[Refer sample input & output ].

**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 First Customer Details :

Enter Customer Name :

**Roger**

Enter Customer Email :

**abc@xyz.com**

Enter Second Customer Details :

Enter Customer Name :

**Lee**

Enter Customer Email :

**abc@xYz.com**

The Email ids of Roger and Lee are not equal

Comparing without considering the cases :

The Email ids of Roger and Lee are Equal

Top of Form

Bottom of Form

**Date - 2**

Neerja Banhot was the head of the all Indian cabin crew in the Pan Am 73 flight. Neerja made sure that everything inside the flight was fine.



She noticed that the date being displayed inside the flight in an LED Board was wrong.



She asked help from a few electronics engineers who were on board. The electronics engineers figure out that the binary forms of the  date, month and year were 2 bits, 2 bits and 2 bits rotated left. Now the engineers will need to fix this. Given the incorrect date, month and the year use the Integer Wrapper Class **rotateRight()** method to print the correct date in International format.  
  
**Input Format:**  
The first line is an integer that corresponds to the incorrect day number.  
The second line is an integer that corresponds to the incorrect month number.  
The third line is an integer that corresponds to the incorrect year.  
  
**Output Format:**  
The Output should display the correct date in a single line separated by slashes in the international format.  
  
**Sample Input:**  
20  
36  
7944

**Sample Output:**  
1986/9/5

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form

Bottom of Form

}Top of Form

Bottom of Form

Top of Form

Bottom of Form

Top of Form

Bottom of Form