### S1P13-PRANAV AND CHANGE

**Pranav and Change**

Pranav, an enthusiastic kid visited the "Fun Fair 2017" along with his family. His father wanted him to purchase entry tickets from the counter for his family members. Being a little kid, he is just learning to understand about units of money. Pranav has paid some amount of money for the tickets but he wants your help to give him back the change of Rs.**N**using minimum number of rupee notes.  
   
Consider a currency system in which there are notes of seven denominations, namely, Rs. 1, Rs. 2, Rs. 5, Rs. 10, Rs. 50, Rs. 100. If the change given to Pranav Rs. **N** is input, write a program to compute smallest number of notes that will combine to give Rs. **N.**  
  
**Note:**  
Refer to problem specifications.  
  
**Input Format:**  
First line of the input is an integer N, the change to be given to Pranav.  
   
**Output Format:**  
Output should display the the smallest number of notes that will combine to give **N**.  
Refer sample input and output for formatting specifications.  
  
**Sample Input 1:**  
1200  
  
**Sample Output1:**  
12  
  
**Sample Input 2:**  
242  
  
**Sample Output2:**  
7

**import java.util.Scanner;**

**class Main {**

**public static void main(String[] args) {**

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

**int num = sc.nextInt();**

**int count = 0;**

**while( num >= 100)**

**{**

**num -= 100;**

**count++;**

**}**

**while( num >= 50)**

**{**

**num -= 50;**

**count++;**

**}**

**while( num >= 10)**

**{**

**num -= 10;**

**count++;**

**}**

**while( num >= 5)**

**{**

**num -= 5;**

**count++;**

**}**

**while( num >= 2)**

**{**

**num -= 2;**

**count++;**

**}**

**while( num >= 1)**

**{**

**num -= 1;**

**count++;**

**}**

**System.out.println(count);**

**}**

**}**

**Co-Partners in Train**

Tim and Bob are off to a famous Education Fair "Knowledge Forum 2017" at Uzhlanda. This time they have to travel without their guardians. Tim got very interested in the arrangement of seats inside the train coach.  
The entire coach could be viewed as an arrangement of consecutive blocks of size 8.

|  |  |
| --- | --- |
| **BerthNumber** | **Compartment** |
| 1-8 | 1 |
| 9-16 | 2 |
| 17-24 | 3 |
| ... and so on |  |

Each of these size-8 blocks are further arranged as:

|  |
| --- |
| 1LB,  2MB,  3UB,  4LB,  5MB,  6UB,  7SL,  8SU |
| 9LB, 10MB, ... |
| ....... |
| ....... |

Here LB denotes lower berth, MB middle berth and UB upper berth.  
   
The following berths are called Co-Partners in Train:

|  |  |
| --- | --- |
| 3 UB | 6 UB |
| 2 MB | 5 MB |
| 1 LB | 4 LB |
| 7 SL | 8 SU |

and the pattern is repeated for every set of 8 berths.   
Tim and Bob are playing this game of finding the co-partner in train of each berth. Write a program to do the same.  
   
**Input Format:**  
The input consists of an integer N, which corresponds to the berth number whose neighbor is to be found out.  
  
**Output Format:**  
The output is to display the berth of the neighbor of the corresponding seat.  
Refer sample input and output for formatting specifications.  
  
**Sample Input 1:**  
1  
  
**Sample Output 1:**  
4LB  
  
**Sample Input 2:**  
5  
  
**Sample Output 2:**  
2MB

import java.util.Scanner;

public class Main{

public static void main(String[] args){

//Fill your code

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

if(n%8>0&&n%8<7)

{

int r=n%8;

if(r<=3)

{

n=n+3;

if(n%8==4)

{

System.out.println(n+"LB");

}

else if(n%8==5)

{

System.out.println(n+"MB");

}

else

{

System.out.println(n+"UB");

}

}

else

{

n=n-3;

if(n%8==1)

{

System.out.println(n+"LB");

}

else if(n%8==2)

{

System.out.println(n+"MB");

}

else

{

System.out.println(n+"UB");

}

}

}

else

{

int r=n%8;

if(r==7)

{

n=n+1;

System.out.println(n+"SU");

}

else

{

n=n-1;

System.out.println(n+"SL");

}

}}

}

**Hollow Pyramid**

The much awaited event at the entertainment industry every year is the "Screen Awards". This year the event is going to be organized on December 25 to honour the Artists for their professional excellence in Cinema. The Organizers of the event, J&R Events, decided to design some attractive and LED Matrix panel boards for the show promotions all across the venue.  
   
The Event organizers wanted to program the display boards with some specific pattern using alphabets and special characters. Help them write a program to design the pattern of a hollow pyramid in the matrix panel, given the number of lines of the pattern.  
   
**Input Format:**  
First line of the input is an integer that refers to the number of lines in the pattern.  
  
**Output Format:**  
Output the pattern as given in the output.  
Refer sample input and output for formatting specifications.  
  
**Sample Input 1:**  
4  
  
**Sample Output 1:**  
bbb\*bbb  
bb\*i\*bb  
b\*iii\*b  
\*\*\*\*\*\*\*  
  
**Sample Input 2:**  
5  
  
**Sample Output 2:**  
bbbb\*bbbb  
bbb\*i\*bbb  
bb\*iii\*bb  
b\*iiiii\*b  
\*\*\*\*\*\*\*\*\*

Top of Form

Bottom of Form

**import java.util.\*;**

**class Main**

**{**

**public static void main(String[] args)**

**{**

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

**int num=sc.nextInt();**

**int si = num+(num-1);**

**int sp = num;**

**int st = -1;**

**for(int i=1;i<=num;i++)**

**{**

**sp = sp-1;**

**st = st+2;**

**for(int j=1;j<=sp;j++)**

**{**

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

**}**

**for(int k=1;k<=st;k++)**

**{**

**if(i>1 && i<num)**

**{**

**if(k>1 && k<st)**

**{**

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

**}**

**else**

**{**

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

**}**

**}**

**else**

**{**

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

**}**

**}**

**for(int l=1;l<=sp;l++)**

**{**

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

**}**

**System.out.println();**

**}**

**}**

**}**

### AF\_PATTERN5

Write a program to print the below pattern.  
  
If n = 3, pattern will be  
  
1 1 1 2  
3 2 2 2  
3 3 3 4  
  
**Input Format:**  
Input consists of an integer n.  
  
**Output Format:**  
Print the desired pattern.  
  
**Sample Input:**  
3  
**Sample Output:**  
1 1 1 2  
3 2 2 2  
3 3 3 4  
  
**Sample Input:**  
4  
**Sample Output:**  
1 1 1 1 3  
4 2 2 2 2  
3 3 3 3 5  
6 4 4 4 4

import java.util.Scanner;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int i=1;

int j=1;

if(i%2!=0)

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

{

if(i%2!=0)

{

for(j=1;j<=n;j++)

{

System.out.print(i+" ");

}

if(j==n+1)

System.out.print(n+i-2);

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

}

else

{

System.out.print(n+i-2);

for(j=2;j<=n+1;j++)

{

System.out.print(" "+i);

}System.out.print("\n");

}

}

//Fill your code

}

}

### CUSTOMER CLASS WITH CONSTRUCTOR

**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

public class Customer{

//Fill your code

public String customerName;

public String customerEmail;

public String customerType;

public String customerAddress;

Customer(String customerName,String customerEmail,String customerType,String customerAddress){

this.customerName=customerName;

this.customerEmail=customerEmail;

this.customerType=customerType;

this.customerAddress=customerAddress;

}

/\* public String getCustomerName() {

return customerName;

}

public void setCustomerName(String customerName) {

this.customerName = customerName;

}

public String getCustomerEmail() {

return customerEmail;

}

public void setCustomerEmail(String customerEmail) {

this.customerEmail = customerEmail;

}

public String getCustomerType() {

return customerType;

}

public void setCustomerType(String customerType) {

this.customerType = customerType;

}

public String getCustomerAddress() {

return customerAddress;

}

public void setCustomerAddress(String customerAddress) {

this.customerAddress = customerAddress;

}\*/

public void displayDetails(){

//Fill your code

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

System.out.println("E-mail: "+customerEmail);

System.out.println("Type: "+customerType);

System.out.println("Location: "+customerAddress);

}

}

import java.io.\*;

import java.util.\*;

public class Main {

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

//Fill your code

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

System.out.println("Enter the Customer Details");

Scanner sc=new Scanner(System.in);

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

String name=sc.next();

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

String email=sc.next();

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

String type=sc.next();

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

String loc=sc.next();

Customer cs=new Customer(name,email,type,loc);

cs.displayDetails();

}

}

### SUM OF AN ARRAY

**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 |

import java.util.\*;

public class Main{

public static void main (String[] args){

//Fill your code

Scanner sc =new Scanner(System.in);

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

int n = sc.nextInt();

int i;

int sum=0;

if(n<=15){

int arr[] = new int[n];

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

arr[i] =sc.nextInt();

}

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

sum = sum +arr[i];

}

System.out.println("Sum of array elements is : "+sum);

}

}

}

### STRING METHODS - BASIC

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

import java.util.\*;

public class Main{

public static void main (String[] args){

//Fill your code

Scanner sc = new Scanner(System.in);

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

String str1 = sc.nextLine();

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

String str2 = sc.nextLine();

System.out.println("Upper Case : "+str1.toUpperCase());

System.out.println("Lower Case : "+str2.toLowerCase());

}

}

Top of Form

Bottom of Form

**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

class Main {

public static void main(String[] args) {

// Fill your code here

System.out.println(args[0]+" - Command Line Arguments");

}

}Top of Form

Bottom of Form

**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

**Best Mobile Plan**

St. Patrick Convent organizes a project exhibition "Innovative Minds" every year with an objective to provide the platform and unleash the potential of the students by showcasing their innovative projects.  
Albert is a science expert and is a topper at his high school. He became interested about the project exhibition and enrolled his name for the same.  
   
Albert’s Dad has a cell phone but often seemed to worry about the price plans for his phone that best fits for his usage pattern and monthly expenses. There are two options, each plan has different costs for daytime minutes, evening minutes and weekend minutes.

**[Note: 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 separate classes in separate files.]**

|  |  |  |  |
| --- | --- | --- | --- |
| **Plan** | **Costs** | | |
| daytime | evening | weekend |
| A | 100 free minutes then 25 rupees per minute | 15 rupees per minute | 20 rupees per minute |
| B | 250 free minutes then 45 rupees per minute | 35 rupees per minute | 25 rupees per minute |

Having this as a spark for his project, Albert decided to design a handy application that will input the number of each type of minutes and output the cheapest plan for this usage pattern,using the format shown below. In the case that the two plans are the same price, output both plans.  
   
He needs your help to evaluate his project and suggest corrections.  
  
Hence create a class named **BestMobilePlan** with the following method.

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| void printPlanDetails(int day, int night, int weekend) | This method should display the cheapest plan for this usage pattern. |

Create another class **Main**and write a main method to test the above class.  
  
In the main method, obtain input from the user in the console and call the printPlanDetails method present in the BestMobilePlan class.

**Input Format:**

First line of the input is an integer that gives the usage during the daytime in minutes.  
Second line of the input is an integer that gives usage during the evening in minutes.  
Third line of the input is an integer that gives usage during the night in minutes.  
  
**Output Format:**  
Output should print the cheapest plan for this usage pattern. In the case that the two plans are the same price, output both plans.  
Refer sample input and output for formatting specifications.  
  
**Sample Input 1:**  
251  
10  
60  
  
**Sample Output 1:**  
Plan A costs 51.25  
Plan B costs 18.95  
Plan B is cheapest  
  
**Sample Input 2:**  
162  
61  
66  
  
**Sample Output 2:**  
Plan A costs 37.85  
Plan B costs 37.85  
Plan A and B are the same price

**Junior Coders**

Junior Coders Academy is a unique learning Centre that offers a creative and inspiring collaborative environment for building coding skills to students of Grades 1 to 12.  
Williams, the proprietor of the Academy and the mentor for the students started his first session of the day with the interesting programming concept of using Functions. Post an interactive session of learning through design, Williams gave the students a small self-activity to verify from two integer numbers A and B, if B corresponds to the last digit/digits of A. Williams wanted you to write the program for evaluating the students’ codes.  
  
Hence create a class named **EvaluateCodes** with the following method.

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| int findValue(String,String) | This method should return 1 if the second number B is the last digits of A, else return 0. |

Create another class **Main**and write a main method to test the above class.  
  
In the main method, obtain input from the user in the console and display "Yes" if the second number B is the last digits of A, else Print "No" by calling the findValue method present in the EvaluateCodes class.

**[Note: 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 separate classes in separate files.]**

**Input Format:**  
First line of the input contains the number**A.**  
Second line of the input contains the number **B.**  
  
**Output Format:**  
Print "Yes"(without quotes) if the second number B is the last digits of A. Print "No"(without quotes) otherwise.  
Refer sample input and output for formatting specifications.  
  
**Sample Input 1:**  
1234  
1234  
  
**Sample Output 1:**  
Yes  
  
**Sample Input 2:**  
5434554  
543  
  
**Sample Output 2:**  
No

**SPELL BEE CONTEST**

Given a pair of words (the first is the correct spelling and the second is the contestant’s spelling of the word) determine if the word is spelt correctly.

The degree of correctness is as follows:

* CORRECT if it is an exact match
* ALMOST CORRECT if no more than 2 letters are wrong
* WRONG if 3 or more letters are wrong

**Input Format:**

Input consists of a pair of words.

**Output Format:**

Output the contestant’s spelling of the word and the degree of correctness. All letters are upper case. The output is to be formatted exactly like that for the sample output given below.

**Assumptions:**  
Words contain only upper case letters. The maximum word length is 10.

**Sample Input:**  
SAMPLE  
SIMPLE  
  
**Sample Output:**  
SIMPLE IS ALMOST CORRECT

import java.io.\*;

import java.util.\*;

class EvaluateCodes{

int findValue(String a, String b)

{

//Fill your code

char[] c=a.toCharArray();

char[] d=b.toCharArray();

int temp=c.length-1;

int count=0;

if(c.length>=d.length)

{

for(int i=d.length-1;i>=0;i--)

{

if(d[i]==c[temp])

{

count++;

temp--;

}

}

}

if(count==d.length)

return 1;

else

return 0;

}

}

public class Main {

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

//Fill your code

Scanner s=new Scanner(System.in);

String a=s.nextLine();

String b=s.nextLine();

EvaluateCodes e=new EvaluateCodes();

if(e.findValue(a,b)==1)

{

System.out.println("Yes");

}

if(e.findValue(a,b)==0)

System.out.println("No");

}

}

### COMMAND LINE ARGUMENTS II

**Command Line Arguments II**

A Java application can accept any number of arguments from the command line.  
  
Write a program to add 2 integers passed as command line arguments and to print their sum. If the number of command line arguments passed is not equal to 2, then print Invalid Input.  
  
**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:  
//Type this in command line. Assume that the class name is Main  
java Main 5 6**  
  
The sum of 5 and 6 is 11  
  
**Sample Input and Output 2:  
//Type this in command line. Assume that the class name is Main**  
**java Main**  
  
Invalid Input  
  
**Sample Input and Output 3:  
//Type this in command line. Assume that the class name is Main  
java Main 1 2 3**  
  
Invalid Input

public class Main {

public static void main(String[] args) {

// TODO Auto-generated method stub

int a=0,b=0,c=0;

try

{

if(args.length>0 && args.length<=2)

{

a=Integer.parseInt(args[0]);

b=Integer.parseInt(args[1]);

c=a+b;

System.out.println("The sum of "+args[0]+" and "+args[1]+" is "+c);

}

else

{

throw new ArrayIndexOutOfBoundsException();

}

}

catch(ArrayIndexOutOfBoundsException e)

{

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

}

}

}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

### EVENT DETAILS -SIMPLE OVERRIDE

**Event Details -Simple Override**

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

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

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| String | name |
| String | detail |
| String | organizer |

Include a three-argument constructor with parameters in the following order,   
**public Event(String name, String detail, String organizer)**  
  
In the **Event**class include the following method

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void totalCredit() | Display "Credit Details" |

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

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| int | stallCount |

Include a four-argument constructor with parameters in the following order,    
**public Exhibition(String name, String detail, String organizer, int stallCount)**  
  
Use **super**Keyword to call the base class constructor.   
  
In the **Exhibition**class override the **totalCredit()** method of the **Event** Class

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void totalCredit() | Call TotalCredit() of base class using **base.** Compute the total credit and display the same |

**Formula to calculate total credits for exhibition:**(stallCount \* 50)  
  
Override **toString()** method in the **Exhibition**class to display the Exhibition Details.  
  
Create class **StageEvent**with the following private attributes/variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| int | totalShow |
| int | seatsPerShow |

Include a five-argument constructor with parameters in the following order,    
**public StageEvent(string \_name, string \_detail, string \_organizer, int \_totalShow, int \_seats)**  
  
In the **StageEvent**class override the **totalCredit()** method of the **Event** Class

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void totalCredit() | Call TotalCredit() of base class using base.Compute the total credit and display the same |

**Formula to calculate total credits for stage event:**  
(totalShow\* seatsPerShow )\* 100  
  
Override **toString()** method in the **StageEvent**class to display the stage event details.  
  
Create **Main** class with **main** method, read the Event type from the user and create corresponding objects.  
Display Event Details along with total credit.  
  
**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

**The phenomenal show**

Enter Detail

**World wide best stage show activites**

 Enter Organizer

**john**

Select Event Type 1.Exhibition 2.StageEvent

**2**

Enter Total shows

**5**

Enter seats per show

**100**

Event Name : The phenomenal show

Event Detail : World wide best stage show activites

Event Organizeer : john

Total Events : 5

Total Seats : 100

Credit Details

Total Credit Gained is 50000

**Sample Input and Output 2:**  
 Enter Event Name

**Dubai Food fest**

Enter Detail

**dubai food items**

Enter Organizer

**faizur**

Select Event Type 1.Exhibition 2.StageEvent

**1**

Enter stall count

**100**

Event Name : Dubai Food fest

Event Detail : dubai food items

Event Organizer : faizur

Stall Count : 100

Credit Details

Total Credit Gained is 5000

**Sample Input and Output 3:**

Enter Event Name

**sevens football**

Enter Detail

**football tournament**

Enter Organizer

**gowtham**

Select Event Type 1.Exhibition 2.StageEvent

**5**

Enter valid choice

Top of Form

Bottom of Form

### STUDENT-FEEDBACK(SINGLE INHERITANCE)

**Student-Feedback(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. ]**  
   
Create a class named **Student** with the following  member variables / attributes  .

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| Integer | id |
| String | name |
| String | department |
| Integer | courseId |

Include a 4-argument constructor. The order of parameters passed to the constructor is id,name,department,courseId.  
Override **toString( )** method to display the details of the class.  
   
Create a child class named **StudentRating** which extends **Student**with the following  member variables / attributes  .

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| Integer | id |
| String | review |
| Integer | stars |
| Student | student |

Include a 4-argument constructor. The order of parameters passed to the constructor is id,review, stars, student(inherited from Student class.  
Override **toString( )** method to display the details of the 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 student id  
**12**  
Enter the student name  
**Prakash**  
Enter the department  
**ECE**  
Enter the course id  
**250**  
Enter the Rating id  
**4**  
Enter review  
**Very good Student!!!**  
Enter number of stars  
**5**  
Student :  
Id :  12  
Name :  Prakash  
Department :  ECE  
Course Id :  250  
Rating ID :  4  
Review :  Very good Student!!!  
Rating Stars :  5

Top of Form

Bottom of Form

### OVERLOADING MAKEPAYMENT()

**Overloading makePayment()**

Consider doing an extra feature for the stage show organisers. Bring up an interactive console application for Billing so that our application looks unique from other competitors. Customers pay using cash, online wallets, and credit card. For each category obtain necessary information from the user. You also require a receipt for all the transactions which should be printed at the end of the transaction. Let's increase our coding proficiency by implement Function overloading for the payments. Hence write a program meeting all the above specification.  
  
Create a class named **TicketBooking** with the following private attributes.

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| stageEvent | String |
| customer | String |
| noOfSeats | Integer |

Include getters and setters for the class.  
Include default and parameterized constructors.  
Format for parameterized constructor is **TicketBooking(String stageEvent, String customer, Integer noOfSeats)**  
  
The TicketBooking class has the following methods.

|  |  |
| --- | --- |
| Method Name | Description |
| public void makePayment(Double amount) | This method is for cash payment. This method accepts amount as input and displays the transaction detail |
| public void makePayment(String walletNumber ,Double amount) | This method is for wallet payment. This method accepts amount and wallet number as input and displays the transaction detail. |
| public void makePayment(String creditCard,String ccv,String name,Double amount) | This method is for credit card payment. This method accepts credit card detail, ccv, card holder name ,and amount as input and displays the transaction detail |

Create a driver class called Main. In the Main method, obtain input from the user in CSV format and call appropriate methods for transactions. If choice other than specified is chosen, print "Invalid choice".  
Note: display one digit after decimal point for double values.  
Format for TicketBooking Input is **stageEvent,customer,noOfSeats**  
**[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.]**  
  
**[All text in bold corresponds to the input and rest corresponds to the output]**  
**Sample Input/Output 1:**  
  
Enter the Booking details  
**Magic show,Mahesh,5**  
Payment mode  
1.Cash payment  
2.Wallet payment  
3.Credit card payment  
**1**  
Enter the amount  
**500**  
Stage event:Magic show  
Customer:Mahesh  
Number of seats:5  
Amount 500.0 paid in cash  
  
**Sample Input/Output 2:**  
  
Enter the Booking details  
**Motivational speech,Rajesh,10**  
Payment mode  
1.Cash payment  
2.Wallet payment  
3.Credit card payment  
**2**  
Enter the amount  
**400**  
Enter the wallet number  
**AFG-456**  
Stage event:Motivational speech  
Customer:Rajesh  
Number of seats:10  
Amount 400.0 paid using wallet number AFG-456  
  
**Sample Input/Output 3:**  
  
Enter the Booking details  
**Debate,Raja,2**  
Payment mode  
1.Cash payment  
2.Wallet payment  
3.Credit card payment  
**3**  
Enter card holder name  
**Raja**  
Enter the amount  
**200**  
Enter the credit card type  
**Master**  
Enter the CCV number  
**9874-4758-9856**  
Stage event:Debate  
Customer:Raja  
Number of seats:2  
Holder name:Raja  
Amount 200.0 paid using Master card  
CCV:9874-4758-9856

Top of Form

Bottom of Form

### ACCOUNT DETAILS - HCL

**Account Details - HCL**

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

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

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| String | holderName |
| long | accNumber |
| String | IFSCCode |
| long | contactNumber |

Include a four-argument constructor with parameters in the following order,  
**public Account(String holderName, long accNumber, String IFSCCode, long contactNumber)**  
  
The class**Account**should have the following method

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void display() | This method displays account details in the following order holderName, accNumber, IFSCCode and contactNumber |

Create class **SavingAccount**which extends **Account** class with the following private attributes/variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| double | interestRate |

In the **SavingAccount**class include the following method

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void display() | Invokes the base class display() and in addition displays interestRate |

Include five-argument constructor with parameters in the following order,   
**public SavingAccount(String holderName, long accNumber, String IFSCCode, long contactNumber, double interestRate)**  
Use **super**Keyword to call the base class constructor.  
  
Create class **CurrentAccount**which extends **Account** class with the following private attributes/variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| String | orgName |
| long | TIN |

In the **CurrentAccount**class include the following method

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void display() | Invokes base class display() and in addition displays orgName,TIN |

Include a six-argument constructor with parameters in the following order,   
**public CurrentAccount(String holderName, long accNumber, String IFSCCode, long contactNumber, String orgName, long TIN)**  
  
**Note:**Use **super**Keyword to call the base class constructor.  
  
Create **Main** class with **main** method, get user details in comma seperated format in the following order **(HolderName,Account Number,IFSC code,Contact Number) .**  
Display the Account Details by calling method of base class with child class object.  
  
**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 User Details(HolderName,Account Number,IFSC code,Contact Number)

**kavitha,982714210,S160030600514,9092304676**

Enter Account Type

**saving**

Enter Interest Rate

**12.0**

Your Contact Details

HolderName : kavitha

Account Number : 982714210

IFSCCode : S160030600514

ContactNumber : 9092304676

Interest Rate : 12

**Sample Input and Output 2:**

Enter User Details(HolderName,Account Number,IFSC code,Contact Number)

**arun,7889142075,S1600ABY0576,9944001700**

Enter Account Type

**current**

Enter organization Name

**pentamedia Graphics Limited**

Enter TIN number

**7841**

Your Contact Details

HolderName : arun

Account Number : 7889142075

IFSCCode : S1600ABY0576

ContactNumber : 9944001700

Organization Name : pentamedia Graphics Limited

TIN : 7841  
  
**Sample Input and Output 3:**  
Enter User Details(HolderName,Account Number,IFSC code,Contact Number)  
**Ram,987451024,SWQ78914AF,9078425168**  
Enter Account Type  
**curr**  
Enter valid Account Type

Top of Form

Bottom of Form

class SavingAccount extends Account

{

private double interestRate;

public SavingAccount(String holderName, long accNumber, String IFSCCode, long contactNumber, double interestRate)

{

super(holderName,accNumber,IFSCCode,contactNumber);

this.interestRate=interestRate;

}

public double getInterestRate()

{

return this.interestRate=interestRate;

}

public void setInterestRate(double interestRate)

{

this.interestRate=interestRate;

}

public void display( )

{

super.display();

double res = getInterestRate();

int r = (int)getInterestRate();

if((res>=r+(0.01)) && (res<=(r+0.99)))

{

System.out.println("Interest Rate : "+getInterestRate());

}

else

{

System.out.println("Interest Rate : "+(int)getInterestRate());

}

}

}

class Account{

private String holderName;

private long accNumber;

private String IFSCCode;

private long contactNumber;

public Account(String holderName, long accNumber, String IFSCCode, long contactNumber)

{

this.holderName=holderName;

this.accNumber=accNumber;

this.IFSCCode=IFSCCode;

this.contactNumber=contactNumber;

}

public String getHolderName()

{

return this.holderName;

}

public void setHolderName(String holderName)

{

this.holderName=holderName;

}

public long getAccNumber()

{

return this.accNumber;

}

public void setAccNumber(long accNumber)

{

this.accNumber=accNumber;

}

public String getIFSCCode()

{

return this.IFSCCode;

}

public void setIFSCCode(String IFSCCode)

{

this.IFSCCode=IFSCCode;

}

public long getContactNumber()

{

return this.contactNumber;

}

public void setContactNumber(long contactNumber)

{

this.contactNumber=contactNumber;

}

public void display()

{

System.out.println("Your Contact Details");

System.out.println("HolderName : "+getHolderName());

System.out.println("Account Number : "+getAccNumber());

System.out.println("IFSCCode : "+getIFSCCode());

System.out.println("ContactNumber : "+getContactNumber());

}

}

public class CurrentAccount extends Account

{

private String orgName;

private long TIN;

public CurrentAccount(String holderName, long accNumber, String IFSCCode, long contactNumber,String orgName, long TIN)

{

super(holderName,accNumber,IFSCCode,contactNumber);

this.orgName=orgName;

this.TIN=TIN;

}

public String getOrgName()

{

return this.orgName;

}

public void setOrgName(String orgName)

{

this.orgName = orgName;

}

public long getTIN()

{

return this.TIN;

}

public void setTIN(long TIN)

{

this.TIN=TIN;

}

public void display( )

{

super.display();

System.out.println("Organization Name : "+getOrgName());

System.out.println("TIN : "+getTIN());

}

}

import java.io.\*;

import java.util.\*;

public class Main

{

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter User Details(HolderName,Account Number,IFSC code,Contact Number)");

String details=sc.nextLine();

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

long ac =Long.parseLong(str[1]);

long cn=Long.parseLong(str[3]);

System.out.println("Enter Account Type");

String type = sc.nextLine();

if(type.equals("saving"))

{

System.out.println("Enter Interest Rate");

double i=sc.nextDouble();

SavingAccount sa = new SavingAccount(str[0],ac,str[2],cn,i);

sa.display();

}

else if(type.equals("current"))

{

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

String org = sc.nextLine();

System.out.println("Enter TIN number");

long tin = sc.nextLong();

CurrentAccount ca = new CurrentAccount(str[0],ac,str[2],cn,org,tin);

ca.display();

}

else

{

System.out.println("Enter valid Account Type");

}

}

}

### INHERITANCE - BOOKING TICKETS

**Inheritance - Booking Tickets**

**[Strictly adhere to the Object Oriented Specifications mentioned in the problem. Use the same Class names, attribute names and method names specified here. Use appropriate naming conventions for getters and setters. Create all classes in separate files.]**

Whether you fly for business or leisure purposes, you can choose either private jets or commercial airlines to get from point A to point B.

The task is to get the flight details and display.

Write a Java program to Implement this task.

Create a class **Aircraft**

Include the following protected data members / attributes:

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| String | aircraftName |
| String | source |
| String | destination |

Include **getters**/ **setters**.

Include the following public methods

Include a **3 argument constructor**, as shown below

public Aircraft(String aircraftName,String source,String destination)

**displayDetails**– The return type of this method is void, this method prints the details of the Aircraft(aircraftName,source and destination)

Create a class **PublicAircraft** that extends **Aircraft**

Include the following private attributes / data members:

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| Boolean | checkinbeforetwohours |
| int | noOfKgsallowed |
| float | additionalFeeperkg |

 Here checkinbeforetwohours is a Boolean value that says whether the passenger should check in before two hours or not.

 This flag is true in case of PublicAircraft.

Include **getters**/ **setters**.

Include the following public methods

Include a **6 argument constructor**, as shown below

public PublicAircraft(String aircraftName,String source,String destination,Boolean checkinbeforetwohours,int noOfKgsallowed,float additionalFee/kg)

**displayDetails**– The return type of this method is void, this method prints the details of the booking done for the Public aircraft directly(along with the aircraft name,source and destination) refer sample input and output.

Create a class **PrivateAircraft** that extends **Aircraft**

Include the following private attributes / data members:

|  |  |
| --- | --- |
| **Data Type** | **Variable Name** |
| Boolean | checkinbeforetwohours |
| String | pilotPreference |
| String | purpose |

Here checkinbeforetwohours is a Boolean value that says whether the passenger should check in before two hours or not.

 This flag is false in case of PrivateAircraft.

Here, pilot preferences is a string for which the name of the pilot should be given by the passenger according to his preference.

  Purpose is a string that indicates the purpose of the flight (Medical,Personal,Cargo)

Include **getters**/ **setters**.

Include the following public methods

Include a **6 argument constructor**, as shown below

public PrivateAircraft(String aircraftName,String source,String destination,Boolean checkinbeforetwohours,String pilotPreference ,String  purpose )

**displayDetails**- The return type of this method is void, this method prints the details of the booking done by the agency(along with the aircraft name,source and destination) refer sample input and output.

Create a **Main**class to test the classes defined above.

**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 1:**

Enter the name of the Aircraft

**Jet Airways**

Enter the source

**Bangalore**

Enter the destination

**Chennai**

Enter the type of Flight

1.Public Aircraft

2.Private Aircraft

**1**

Is the flight check in before two hours

**Yes**

Enter the number of kgs allowed per person

**15**

Enter the additional fee charged for extra baggage per Kg

**750.00**

Flight Details :

Public Aircraft :

Aircraft Name : Jet Airways

Source : Bangalore

Destination : Chennai

Flight check in before two hours : Yes

Number of kgs allowed per person : 15

Additional fee charged for extra baggage per Kg : 750.00

**Sample Input Output 2:**

Enter the name of the Aircraft

**Spice Jet**

Enter the source

**Bangalore**

Enter the destination

**Chennai**

Enter the type of Flight

1.Public Aircraft

2.Private Aircraft

**2**

Is the flight check in before two hours

**No**

Enter the name of the pilot chose

**Akilan**

Enter the Purpose of your flight

**Medical**

Flight Details :

Private Aircraft :

Aircraft Name : Spice Jet

Source : Bangalore

Destination : Chennai

Flight check in before two hours : No

Pilot chose : Akilan

Purpose of the flight : Medical

 public class PublicAircraft extends Aircraft

{

String b1;

void displayDetails(int kg,double fee,String b1)

{

if(b1.equalsIgnoreCase("yes"))

{

b1="Yes";

}

else

{

b1="No";

}

super.displayDetails();

System.out.println("Flight check in before two hours : "+b1);

System.out.println("Number of kgs allowed per person : "+kg);

System.out.print("Additional fee charged for extra baggage per Kg : ");

System.out.printf("%.2f",fee);

}

}

public class PrivateAircraft extends Aircraft{

String a;

void displayDetails(String a,String b,String c)

{

if(a.equalsIgnoreCase("yes"))

{

a="Yes";

}

else

{

a="No";

}

super.displayDetails();

System.out.println("Flight check in before two hours : "+a);

System.out.println("Pilot chose : "+b);

System.out.println("Purpose of the flight : "+c);

}

}

public class Aircraft

{

String name,source,destination;

public Aircraft()

{

this.name=name;

this.source=source;

this.destination=destination;

}

public String getName()

{

return name;

}

public String getSource()

{

return source;

}

public String getDestination()

{

return destination;

}

public void setName(String name)

{

this.name=name;

}

public void setSource(String source)

{

this.source=source;

}

public void setDestination(String destination)

{

this.destination=destination;

}

void displayDetails()

{

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

System.out.println("Source : "+getSource());

System.out.println("Destination : "+getDestination());

}

}

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main

{

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

{

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

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

String aircraftName=br.readLine();

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

String source=br.readLine();

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

String destination = br.readLine();

System.out.println("Enter the type of Flight\n1.Public Aircraft\n2.Private Aircraft");

int choice=Integer.parseInt(br.readLine());

if(choice==1)

{

System.out.println("Is the flight check in before two hours");

String ans = br.readLine();

Boolean b1;

if(ans.equals("yes"))

{

b1=true;

}

else

{

}

System.out.println("Enter the number of kgs allowed per person");

int noOfKgsallowed=Integer.parseInt(br.readLine());

System.out.println("Enter the additional fee charged for extra baggage per Kg");

float additionalFeeperkg=Float.parseFloat(br.readLine());

PublicAircraft b = new PublicAircraft();

b.setName(aircraftName);

b.setSource(source);

b.setDestination(destination);

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

System.out.println("Public Aircraft:");

b.displayDetails(noOfKgsallowed,additionalFeeperkg,ans);

}

else if(choice==2)

{

System.out.println("Is the flight check in before two hours");

String ans = br.readLine();

System.out.println("Enter the name of the pilot chose");

String pilotPreference=br.readLine();

System.out.println("Enter the Purpose of your flight");

String purpose=br.readLine();

PrivateAircraft b = new PrivateAircraft();

b.setName(aircraftName);

b.setSource(source);

b.setDestination(destination);

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

System.out.println("Private Aircraft:");

b.displayDetails(ans,pilotPreference,purpose);

}

else{

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

}

}

}

Top of Form

Bottom of Form

### ABSTRACT CLASS - INTRODUCTION

**Abstract Class - Introduction**

Abstract classes are declared with a keyword abstract. These classes cannot be instantiated. They can have attributes/methods. They can have normal methods as well as abstract methods. These abstract methods must be implemented in subclasses or else they should also be declared as abstract. Now let's get on with implementing this newly learned concept in a simple program, before going to our 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 **Shape** with abstract method **public abstract Double calculatePerimeter()**  
  
Create a class **Circle** that extends **Shape** with the following private attributes,

|  |  |
| --- | --- |
| **Attribute** | **Datatype** |
| radius | Float |

Include appropriate getters and setters, constructors for the class.  
  
Create a class **Rectangle** that extends **Shape** with the following private attributes,

|  |  |
| --- | --- |
| **Attribute** | **Datatype** |
| length | Float |
| breadth | Float |

Include appropriate getters and setters, constructors for the class.  
  
Create a class **Square** that extends **Shape** with the following private attributes,

|  |  |
| --- | --- |
| **Attribute** | **Datatype** |
| side | Float |

Include appropriate getters and setters, constructors for the class.  
  
Implement the method **calculatePerimeter()** in all the child classes to calculate appropriate perimeters.  
  
**Note**: Use **3.14** for **pi**value  
  
**Input and Output format:**  
Display the double value correct to 2 decimal place  
Refer to sample Input and Output for formatting specifications.  
  
**[All Texts in bold corresponds to the input and rest are output]**  
**Sample Input and Output 1:**  
  
List of Shapes:  
1.Circle  
2.Rectangle  
3.Square  
Enter your choice:  
**1**  
Enter the radius of the Circle:  
**2.34**  
The perimeter is 14.70  
  
**Sample Input and Output 2:**  
  
List of Shapes:  
1.Circle  
2.Rectangle  
3.Square  
Enter your choice:  
**2**  
Enter the length of the Rectangle:  
**12**  
Enter the breadth of the Rectangle:  
**3**  
The perimeter is 30.00  
  
**Sample Input and Output 3:**  
  
List of Shapes:  
1.Circle  
2.Rectangle  
3.Square  
Enter your choice:  
**3**  
Enter the side of the Square:  
**13**  
The perimeter is 52.00

### public abstract class Shape {

### //write your code here

### public abstract Double calculatePerimeter();

### }

public class Square extends Shape{

//write your code here

private float side;

public Square() {

super();

}

public Square(float side) {

super();

this.side = side;

}

public float getSide() {

return side;

}

public void setSide(float side) {

this.side = side;

}

@Override

public Double calculatePerimeter() {

// TODO Auto-generated method stub

return (double) (4\*getSide());

}

}

public class Circle extends Shape{

//write your code here

private float radius;

public float getRadius() {

return radius;

}

public void setRadius(float radius) {

this.radius = radius;

}

public Circle(float radius) {

super();

this.radius = radius;

}

public Circle() {

super();

}

public Double calculatePerimeter()

{

//System.out.println("Enter the radius of the circle");

//System.out.println("The perimeter is "+2\*3.14\*getRadius());

return 2\*3.14\*getRadius();

}

/\*public String toString() {

return "The perimeter is "+calculatePerimeter();

}\*/

}

public class Rectangle extends Shape {

//write your code here

private float length;

private float breadth;

public Rectangle() {

super();

}

public Rectangle(float length, float breadth) {

super();

this.length = length;

this.breadth = breadth;

}

public float getLength() {

return length;

}

public void setLength(float length) {

this.length = length;

}

public float getBreadth() {

return breadth;

}

public void setBreadth(float breadth) {

this.breadth = breadth;

}

public Double calculatePerimeter()

{

return (double) (2\*(getLength()+getBreadth()));

}

}

import java.util.\*;

public class Main {

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

//write your code here

Scanner sc=new Scanner(System.in);

System.out.println("List of Shapes:");

System.out.println("1.Circle\n" +

"2.Rectangle\n" +

"3.Square");

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

int n=sc.nextInt();

if(n==1)

{

System.out.println("Enter the radius of the Circle:");

float r=sc.nextFloat();

Circle c= new Circle(r);

System.out.printf("The perimeter is %.2f",c.calculatePerimeter());

//c.calculatePerimeter();

}

else if(n==2)

{

System.out.println("Enter the length of the Rectangle:");

float l=sc.nextFloat();

System.out.println("Enter the breadth of the Rectangle:");

float b=sc.nextFloat();

Rectangle rect=new Rectangle(l,b);

System.out.printf("The perimeter is %.2f",rect.calculatePerimeter());

//rect.calculatePerimeter();

}

else if(n==3)

{

System.out.println("Enter the side of the Square:");

float s=sc.nextFloat();

Square sq=new Square(s);

System.out.printf("The perimeter is %.2f",sq.calculatePerimeter());

//sq.calculatePerimeter();

}

}

}

### MULTIPLE INHERITANCE WITH INTERFACES

**Multiple Inheritance with interfaces**

The Interface defines a rule that any classes that implement it should override all the methods. Let's implement Interface in our 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 interface **BikeSpeed**with the following method

|  |  |
| --- | --- |
| **Method** | **Description** |
| public int averageSpeed( ) | abstract method |

Create an interface **BikeDistance**with the following method

|  |  |
| --- | --- |
| **Method** | **Description** |
| public int totalDistance( ) | abstract method |

Create a class **Bike**which implements **BikeSpeed**and**BikeDistance**interfaces with the following private attributes

|  |  |
| --- | --- |
| **Attribute** | **Datatype** |
| distance | int |
| speed | int |

Create default constructor and a parameterized constructor with arguments in order **Bike(int distance, int speed)**.  
Include appropriate getters and setters.  
  
Include the following method in the **Bike**class

|  |  |
| --- | --- |
| **Method** | **Description** |
| public int averageSpeed( ) | This method is used to calculate the average speed maintained for the bike and returns the average speed |
| public int totalDistance( ) | This method is used to calculate the total distance that the bike travelled and returns the total distance |

Create a class named **Main** with the main( ) method to test the above class.  
  
**Note:  Refer sample input and output for specifications**  
**All text in bold corresponds to the input and rest are output  
  
Sample Input and Output:**  
Enter the distance travelled :   
**100**  
Enter the speed of the vehicle :   
**60**  
Total distance travelled : 6000  
Average speed maintained : 100

### STATIC INNER CLASS-2

**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 **Ticket**with the following **public static**attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| noOfSeats | Integer |

Create a static inner class **Platinum**which have the outer class **Ticket.**  
Include the following method in the **Platinum**class

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public Double computeCost() | Here Calculate and return the total cost of the tickets by accessing the attribute noOfSeats of Ticket class. Cost per ticket = Rs.210.50 |

Create a static inner class **Gold**which have the outer class **Ticket.**  
Include the following method in the **Gold**class

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public Double computeCost() | Here Calculate and return the total cost of the tickets by accessing the attribute noOfSeats of Ticket class. Cost per ticket = Rs.168.45 |

Create a static inner class **Silver**which have the outer class **Ticket.**  
Include the following method in the **Silver**class

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public Double computeCost() | Here Calculate and return the total cost of the tickets by accessing the attribute noOfSeats of Ticket class. Cost per ticket = Rs.107.37 |

Get the option for the Ticket type to compute the total cost of the tickets and read the attribute noOfSeats and set the value to the Ticket class attribute. Calculate the total cost and print the same.  
While printing round off the cost 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 type of Ticket.  
The next line of input is total no. of seats.  
  
**Output Format**  
The output consists total cost of the tickets.  
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 ticket type  
1.Platinum  
2.Gold  
3.Silver  
**1**  
Enter the no. of seats  
**2**  
Cost of the tickets is Rs.421.00

**Sample Input/Output 2:**  
Enter the ticket type  
1.Platinum  
2.Gold  
3.Silver  
**2**  
Enter the no. of seats  
**2**  
Cost of the tickets is Rs.336.90

**Sample Input/Output 3:**  
Enter the ticket type  
1.Platinum  
2.Gold  
3.Silver  
**3**  
Enter the no. of seats  
**2**  
Cost of the tickets is Rs.214.74  
  
**Sample Input/Output 4:**  
Enter the ticket type  
1.Platinum  
2.Gold  
3.Silver  
**4**  
Invalid choice

Top of Form

Bottom of Form

### NESTED CLASSES

**Nested classes**

Write a Java program to display stall details by implementing Non-static nested classes concept  
  
Create an outer class **Stall**with following public attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| name | String |
| detail | String |
| owner | String |
| cost | Integer |

Create a parameterized constructor with arguments in order **Stall(String name, String detail, String owner, Integer cost)**.  
  
Create an inner class **GoldStall**which have outer class **Stall**with following public attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| tvSet | Integer |

Create default constructor and a parameterized constructor with arguments in order **GoldStall (Integer tvSet).**  
  
Create an inner class **PlatinumStall**which have outer class **GoldStall** with following public attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| projector | Integer |

Create default constructor and a parameterized constructor with arguments in order **PlatinumStall(Integer projector).**  
  
Include the following method in the **PlatinumStall**Class

|  |  |
| --- | --- |
| **Method name** | **Method Description** |
| public void display() | This method will display the stall details as given in the sample input and output format. |

**Note :**Total cost should be computed by the sum of stall cost, TV set cost and projector cost where each TV set and projector costs 100Rs and 500Rs respectively.  
  
Create a driver class named **Main** to test the above class. Read the stall inputs in comma separated format and split them to set values to the class variables.  
  
**[Note: 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.]  
  
Input Format:**  
The line of input corresponds to the details of the stalls in CSV format.  
  
**Output Format**  
The output consists of stall details.  
Refer sample output for formatting specifications.  
  
**Sample Input/Output:**  
Enter Stall details in comma separated(Stall Name,Stall Description,Owner Name,Stall Cost,Number of TV set,Number of Projectors)  
**Vehicular,Engines are the best,Raizak,6000,5,3**  
Stall Name:Vehicular  
Details:Engines are the best  
Owner Name:Raizak  
TV Sets:5  
Projectors:3  
Total Cost:8000

Top of Form

Bottom of Form

### NESTED CLASSES

**Nested classes**

Write a Java program to display stall details by implementing Non-static nested classes concept  
  
Create an outer class **Stall**with following public attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| name | String |
| detail | String |
| owner | String |
| cost | Integer |

Create a parameterized constructor with arguments in order **Stall(String name, String detail, String owner, Integer cost)**.  
  
Create an inner class **GoldStall**which have outer class **Stall**with following public attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| tvSet | Integer |

Create default constructor and a parameterized constructor with arguments in order **GoldStall (Integer tvSet).**  
  
Create an inner class **PlatinumStall**which have outer class **GoldStall** with following public attributes

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| projector | Integer |

Create default constructor and a parameterized constructor with arguments in order **PlatinumStall(Integer projector).**  
  
Include the following method in the **PlatinumStall**Class

|  |  |
| --- | --- |
| **Method name** | **Method Description** |
| public void display() | This method will display the stall details as given in the sample input and output format. |

**Note :**Total cost should be computed by the sum of stall cost, TV set cost and projector cost where each TV set and projector costs 100Rs and 500Rs respectively.  
  
Create a driver class named **Main** to test the above class. Read the stall inputs in comma separated format and split them to set values to the class variables.  
  
**[Note: 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.]  
  
Input Format:**  
The line of input corresponds to the details of the stalls in CSV format.  
  
**Output Format**  
The output consists of stall details.  
Refer sample output for formatting specifications.  
  
**Sample Input/Output:**  
Enter Stall details in comma separated(Stall Name,Stall Description,Owner Name,Stall Cost,Number of TV set,Number of Projectors)  
**Vehicular,Engines are the best,Raizak,6000,5,3**  
Stall Name:Vehicular  
Details:Engines are the best  
Owner Name:Raizak  
TV Sets:5  
Projectors:3  
Total Cost:8000

Top of Form

Bottom of Form

### EVENTTYPEDOESNOTEXISTSEXCEPTION

**EventTypeDoesNotExistsException**

Now we know how to create a custom exception that suits our needs. So let's try it in a part of our application. All the events should be of the predefined event type. So hardcode some event types before and check if the event entered having the same id or throw EventTypeDoesNotExistsException, and ask the user to reenter the event typeId until they got it right.  
  
Create a class **Event** with the following private attributes,

|  |  |
| --- | --- |
| **Attribute** | **Datatype** |
| name | String |
| detail | String |
| ownerName | String |
| typeId | Long |

Create a class **EventType** with following private attributes,

|  |  |
| --- | --- |
| **Attribute** | **Data type** |
| id | Long |
| name | String |

Include appropriate getters/setters, default and parameterized constructors and override toString() to display the details.  
  
Hard code few event types in driver class **Main**- (Stage event with id-1),(Exhibition with id-2),(Sports meet with id-3).  
  
Create the following static method in the Main class,

|  |  |
| --- | --- |
| **Method** | **Description** |
| public static Boolean isValid(Long typeId,List<EventType> typeList) | This method checks the validity of the event by comparing the type id with the available types. If valid return true, else throw custom Exception. |

Create the custom exception class **EventTypeDoesNotExistsException** that extends Exception.  
  
**Input and Output format:**  
CSV format of the input is **(name,detail,ownerName,typeId)**.  
Use "**%-15s%-15s%-15s%-15s**" to print in tabular form.  
Show message as "**No event type available with the given id**" in Custom exception.  
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 the events:  
**2**  
Enter the details of event 1  
**Book Fair,20% discount,John,2**  
Enter the details of event 2  
**Marathon,50km race,Jane,3**  
The events entered are:  
Name           Details        Owner name     Eventtypeid     
Book Fair      20% discount    John           2                
Marathon       50km race      Jane           3     
  
**Sample Input and Output 2:**  
  
Enter the number of the events:  
**2**  
Enter the details of event 1  
**Book Fair,20% discount,John,2**  
Enter the details of event 2  
**Marathon,50km race,Jane,5**  
EventTypeDoesNotExistsException: No event type available with the given id  
Enter the correct event type id:  
**4**  
EventTypeDoesNotExistsException: No event type available with the given id  
Enter the correct event type id:  
**3**  
The events entered are:  
Name           Details        Owner name     Eventtypeid     
Book Fair      20% discount    John           2                
Marathon       50km race      Jane           3

Top of Form

Bottom of Form

Top of Form

Bottom of Form

**Input Mismatch Exception**

Input Mismatch exception occurs when an input of different datatype is given other than the required. In common practice, it occurs when String or double datatype is given for an int datatype. Let's handle this exception for practice. Obtain int type input from the user using Scanner class. Display the obtained input if no exception occurs. If an exception occurs, prompt the user as specified in Sample Input and Output.  
  
Create a driver class called **Main**. In the Main method, obtain integer input from the user and perform actions as specified above.  
  
**Input and Output format:**  
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 an integer input  
**5**  
Entered value is 5  
  
**Sample Input and Output 2:**  
  
Enter an integer input  
**5.0**  
java.util.InputMismatchException

Top of Form

Bottom of Form

### Problem Requirements:

#### Java

|  |  |  |
| --- | --- | --- |
| Keyword | Min Count | Max Count |
| try | 1 | 1 |

|  |  |  |
| --- | --- | --- |
| Keyword | Min Count | Max Count |
| catch | 1 | 1 |

import java.util.\*;

import java.io.\*;

public class Main {

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

// your code here

Scanner sc=new Scanner(System.in);

try

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

int n=sc.nextInt();

System.out.println("Entered value is "+n);

}

catch(Exception e)

{

System.out.println(e);

}

}

}

### WEAK PASSWORD EXCEPTION

**Weak password Exception**

A typical requirement of a custom exception would be for validation purposes. In this exercise, Let's validate a password input. A password is said to be strong if it satisfies the following criteria  
    i)  It should be a minimum of 10 characters and a maximum of 20 characters.  
    ii) It should contain at least one digit.  
    iii)It should contain at least one special character (non-numeric, non-alphabetic).  
    iv)It should contain at least one letter.  
  
If the password fails any one of the criteria, it is considered as weak.  
  
**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 **User** with the following private attributes.

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| name | String |
| mobile | String |
| username | String |
| password | String |

Include getters and setters.  
Include default and parameterized constructors.  
Format for the parameterized constructor is **User(String name, String mobile, String username, String password)**  
Override the **toString()** method to display the User detail   
  
Create a class called **UserBO**with the following methods.

|  |  |
| --- | --- |
| **Method** | **Description** |
| static void validate(User u) | This method throws WeakPasswordNumber exception if the Password is weak. |

Create a driver class called **Main**. In the Main method, obtain inputs from the user. Validate the password and if there is an exception, handle the exception.  
  
Pass the exception message as "**Your password is weak**".  
  
**Sample Input and Output:**  
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 user details  
**John Doe,9876543210,john,johndoe**  
WeakPasswordException: Your password is weak  
  
**Sample Input and Output 2:**  
  
Enter the user details  
**Jane doe,9876543210,Jane,Janedoe@123**  
Name:Jane doe  
Mobile:9876543210  
Username:Jane  
Password:Janedoe@123

Top of Form

Bottom of Form

**List of List**

We have already seen a problem in the list of lists. So let's try to use it in our application. While the users try to book the tickets for the events they should know the count of remaining tickets. Let's create a list of 5 days of the week each has a list of the count of remaining tickets for 4 shows. List<List<Integer>> is the general format and for the problem, dayList<showList<count>>, ie., store the count of ticket available for each show of a day in a list and then place these lists for each day of a week inside another list.  
  
The maximum number of tickets for a show is 100. So after getting the bulk booked tickets from the user, subtract and store the remaining count of tickets for the whole week in this list of lists.  
  
Create a driver class **Main** and use the main method to get the count of already booked tickets and create a list of the list to store the remaining count.  
  
**Note**:CSV input format is (show1,show2,show3,show4) for each day. And enter day to know remaining ticket count for the day.  
  
Refer sample input/output for other further details and format of the output.  
  
**Input Format:**  
The first five lines have the number of tickets booked in each day  
The next lines have the day in which the remaining ticket to be shown  
  
**[All Texts in bold corresponds to the input and rest are output]  
Sample Input/Output 1:**  
  
Enter the count of booked tickets:  
On Day 1  
**20,25,30,35**  
On Day 2  
**20,20,20,20**  
On Day 3  
**15,25,35,20**  
On Day 4  
**50,60,40,75**  
On Day 5  
**85,88,93,78**  
Enter the day to know its remaining ticket count:  
**5**  
Remaining tickets:[15, 12, 7, 22]  
Do you want to continue?(y/n)  
**y**  
Enter the day to know its remaining ticket count:  
**2**  
Remaining tickets:[80, 80, 80, 80]  
Do you want to continue?(y/n)  
**y**  
Enter the day to know its remaining ticket count:  
**4**  
Remaining tickets:[50, 40, 60, 25]  
Do you want to continue?(y/n)  
**n**

import java.util.\*;

public class Main {

// Create a driver class Main and use the main method

// to get the count of already booked tickets and

// create a list of the list to store the remaining count.

/\*

Input Format:

CSV input format is (show1,show2,show3,show4) for each day.

And enter day to know remaining ticket count for the day.

The first five lines have the number of tickets booked in each day

The next lines have the day in which the remaining ticket to be shown

\*/

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

// While the users try to book the tickets for the events

// they should know the count of remaining tickets.

// Let's create a list of 5 days of the week

// each has a list of the count of remaining tickets for 4 shows.

// List<List<Integer>> is the general format and for the problem,

// dayList<showList<count>>, ie.,

// store the count of ticket available for each show of a day in a list and

// then place these lists for each day of a week inside another list.

// List<Integer>: ticket counts for shows in a day

// List<List<Integer>>: ticket counts for shows of 5 days of the week

List<List<Integer>> ticketsRepo = new ArrayList<>();

for (int i = 0; i < 5; i++) { // initialise one for each day

ticketsRepo.add(new ArrayList<>());

}

// The maximum number of tickets for a show is 100.

final int numDays = 5;

// getting the bulk booked tickets from the user,

// The first five lines have the number of tickets booked in each day

// The next line have the day in which the remaining ticket to be shown

Scanner sc = new Scanner(System.in);

System.out.println("Enter the count of booked tickets:");

for (int di = 1; di <= numDays; di++) {

List<Integer> showTickets = ticketsRepo.get(di-1);

// get show tickets for this day

// subtract and store the remaining count of tickets

System.out.printf("On Day %d%n", di);

String ticketCountStr = sc.next();

String[] tickets = ticketCountStr.split(",");

for (int si = 0; si < tickets.length; si++) {

String ticketStr = tickets[si];

try {

double ticketD = Double.parseDouble(ticketStr);

int ticket = (int) ticketD;

updateShowTickets(showTickets, si, ticket);

} catch (NumberFormatException e) {

// TODO: handle this

}

}

}

// get a day and display its remaining show tickets

String more;

do {

System.out.println("Enter the day to know its remaining ticket count:");

int days = sc.nextInt();

// TODO: validate days

if (days <= numDays) {

List<Integer> showTickets = ticketsRepo.get(days-1);

System.out.printf("Remaining tickets:%s%n", showTickets);

}

System.out.println("Do you want to continue?(y/n)");

more = sc.next();

} while (more.equalsIgnoreCase("y"));

}

/\*\*

\* @modifies: showTickets

\* @effects: update the remaining ticket count of the show at the specified index in showTickets, given

\* the specified bookedTickets.

\*/

private static void updateShowTickets(List<Integer> showTickets, int index, int bookedTickets) {

final int MaxTicketCount = 100;

int remains;

int sz = showTickets.size();

if (sz <= index) {

// extend list to index

for (int i = sz; i <= index; i++) {

showTickets.add(null);

}

}

Integer current = showTickets.get(index);

if (current != null) {

remains = current - bookedTickets;

if (remains < 0) remains = 0;

} else {

remains = 100 - bookedTickets;

}

showTickets.set(index, remains);

}

}

**Generic Methods**

Write a single generic method declaration that can be called with arguments of different types to print the elements of Integer, Double and Character arrays.  
  
**Input Output Format:**  
Input consists of a single integer corresponding to the number of elements in the arrays.  
Refer Sample Input Output for output format.  
  
**Sample Input and Output:**  
Enter a number :  
**6**  
Enter the elements of the integer array  
**1 2 3 4 5 6**  
Enter the elements of the double array  
**1.1 2.2 3.3 4.4 5.5 6.6**  
Enter the elements of the character array  
**a n c d e f**  
Integer array contains:  
1 2 3 4 5 6  
Double array contains:  
1.1 2.2 3.3 4.4 5.5 6.6  
Character array contains:  
a n c d e f

OBSERVATION

2

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.\*;

public class Main {

public static <T> void printArray(T[] d)

{

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

{

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

}

}

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

BufferedReader br = new BufferedReader(

new InputStreamReader(System.in));

Scanner sc = new Scanner(System.in);

System.out.println("Enter a number :");

int n = sc.nextInt();

Integer arr1[] = new Integer[n];

Double arr2[] = new Double[n];

Character arr3[] = new Character[n];

System.out.println("Enter the elements of the integer array");

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

{

arr1[i] = sc.nextInt();

}

System.out.println("Enter the elements of the double array");

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

{

arr2[i] = sc.nextDouble();

}

System.out.println("Enter the elements of the character array");

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

{

arr3[i] = sc.next().charAt(0);

}

System.out.println("Integer array contains:");

printArray(arr1);

System.out.println();

System.out.println("Double array contains:");

printArray(arr2);

System.out.println();

System.out.println("Character array contains:");

printArray(arr3);

}

}

**Seat Arrangement**

Write a Java program to create a structure of seats in a StageEvent given the details of the number of rows and columns and display the no. of seats booked as given in the sample input and output. Assign Section chronologically starting with 'A' and number starting from 1.  
  
**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 **Seat**with following private variables.

|  |  |
| --- | --- |
| **Attributes** | **Datatype** |
| section | Character |
| number | Integer |
| booked | Boolean |

Include **getters**and **setters**.  
Create a **default**and **Parameterized constructors**.  
The format for the parameterized constructor is **Seat(Character section, Integer number,Boolean booked).**  
  
The **Seat**class has the following methods

|  |  |
| --- | --- |
| **Method name** | **Description** |
| static List<List<Seat>> generateSeats(int rows,int seat) | This method gets the number of rows and seats per row as arguments and It returns a List of seat List. |
| static void book(List<List<Seat>> seat,String seats) | This method accepts the List of List of Seats and String containing seats to be booked. It changes the booked variable to true for seats to be booked. |

Create a driver class called **Main**. In the Main method, obtain input from the user and create a list of list of  Seats. obtain Seat details for booking and at last display the Booked seats.  
  
**Input format:**  
The first line corresponds to the number of rows  
The second line corresponds to the number of seats per row  
The third line consists of tickets to be booked in CSV format.  
  
**Output format:**  
Seats that are booked are represented by "**--**" whereas the unbooked seats are represented by the section and number  
  
**[All text in bold corresponds to the input and rest corresponds to the output]  
Sample Input/Output 1:**  
  
Enter the number of rows  
**5**  
Enter the number of seats per row  
**5**  
Enter the seats to book in CSV format  
**A1,B2,C4,D3**

Seats

-- A2 A3 A4 A5

B1 -- B3 B4 B5

C1 C2 C3 -- C5

D1 D2 -- D4 D5

E1 E2 E3 E4 E5 

OBSERVATION

2

import java.util.ArrayList;

import java.util.List;

public class Seat {

private Character section;

private Integer number;

private Boolean booked;

public Seat() {}

public Seat(Character section, Integer number, Boolean booked) {

super();

this.section = section;

this.number = number;

this.booked = booked;

}

public static List<List<Seat>> generateSeats(int rows,int seat){

List<List<Seat>> seats = new ArrayList<>();

Character section='A';

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

List<Seat> l=new ArrayList<>();

Character section1=(char)(section+i);

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

Seat s=new Seat(section1,j,false);

l.add(s);

}

seats.add(l);

}

return seats;

}

public static void book(List<List<Seat>> seat,String seats){

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

for(List<Seat> l : seat)

{

for(int i = 0; i < l.size(); i++)

// for(Seat se : l)

{

int count = 0;

for(String s: str)

{

if(s.charAt(0) == (l.get(i).getSection()) && (int)s.charAt(1) == l.get(i).getNumber()+48)

{

seat.remove(l.get(i));

count++;

}

}

if(count == 0)

{

if(i==l.size()-1)

{

System.out.print(l.get(i).getSection());

System.out.print(l.get(i).getNumber());

}

else {

System.out.print(l.get(i).getSection());

System.out.print(l.get(i).getNumber() + " ");

}

}

else {

if(i==l.size()-1)

{

System.out.print("--");

}

else {

System.out.print("--" + " ");

}

}

}

System.out.println();

}

}

public Character getSection() {

return section;

}

public void setSection(Character section) {

this.section = section;

}

public Integer getNumber() {

return number;

}

public void setNumber(Integer number) {

this.number = number;

}

public Boolean getBooked() {

return booked;

}

public void setBooked(Boolean booked) {

this.booked = booked;

}

}

import java.util.\*;

public class Main {

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

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

int r = sc.nextInt();

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

int s = sc.nextInt();

List<List<Seat>> list = Seat.generateSeats(r,s);

System.out.println("Enter the seats to book in CSV format");

String seats = sc.next();

System.out.println("Seats");

Seat.book(list, seats);

}

}