**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;

}

}