**Inheritance:**

**Discount For Customer**

In the bank, customers can be Normal, Priviledged, SeniorCitizen and so on. The bank also introduces an offer where privileged customers get a 30% off on the bill while senior citizens get 12% off. Let’s implement the inheritance with discount yet again a better understanding.  
  
1. Create Customer, PrivilegedCustomer & SeniorCitizenCustomer class with data members as given below.  
2. Implement GenerateBillAmount Method as per the specification.  
  
Write a program to get the customer details and display bill, discount amount based on customer type.

**[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.]**

Consider a class named **Customer** with the following protected attributes

|  |  |
| --- | --- |
| **Data Type** | **Attributes** |
| string | \_name |
| string | \_address |
| string | \_mobileNumber |
| int | \_age |

The methods for **getters, setters** and **constructors** are given in the template code.

Include the following public method in **Customer**class.

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public void DisplayCustomer() | This method displays the customer details. |

Consider a class **SeniorCitizenCustomer** which extends the class **Customer**.  
  
Include the following public method in **SeniorCitizenCustomer**class.

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| double GenerateBillAmount(int amount) | This method is used to calculate and return the payment amount where the discount is 12%. |

Consider a class **PrivilegeCustomer** which extends the class **Customer**.  
  
Include the following public method in **PrivilegeCustomer**class.

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| double GenerateBillAmount(int amount) | This method is used to calculate and return the payment amount where the discount is 30%. |

Consider a driver class named **Program** which creates an instance of the above mentioned classes and their functionalities are tested.  
Use **base** Keyword to call the base class constructor.  
Read the respective customer details (Senior Citizen or Privileged) and call the corresponding GenerateBillAmount() method based on the choice as shown in the sample output.

**Input and Output Format:**

The bill amount double value should be display 1 decimal palces.

The total amount to be paid value should be displayed upto 2 decimal palces.

Refer sample input and output for formatting specifications.

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

**Sample Input and Output 1:**

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

**Sample Input and Output 2:**

1)Privilege Customer  
2)SeniorCitizen Customer  
Enter Customer Type  
**2**  
Enter The Name  
**Jack**  
Enter The Age  
**46**  
Enter The Address  
**Chennai**  
Enter The Mobile Number  
**7894561230**  
Enter The Purchased Amount  
**500**  
Bill Details  
Name Jack  
Mobile 7894561230  
Age 46  
Address Chennai  
Your bill amount is Rs 500.0. Your bill amount is discount under senior citizen customer  
You have to pay Rs 440.00

**Sample Input and Output 3:**

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

**Account Details**

Write a program to read and display the various type of account details.

**[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.]**

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

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| string | \_holderName |
| long | \_accountNumber |
| string | \_IFSCCode |
| long | \_contactNumber |

Include appropriate **getters** and **setters**.  
Include **default** and **parameterized** constructor for the class.  
Prototype for the Parameterized Constructor **Account(string \_holderName,long \_accountNumber,string \_IFSCCode,long \_contactNumber)**

Define the following method in the **Account**class.

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public void Display() | This method displays account details in the following order \_holderName,\_AccountNumber,\_IFSCCode,\_contactNumber. Display the statement ‘**Your Contact Details**’ inside this method. |

Consider the class **SavingAccount**which inherits **Account** class with the following private attributes/variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| double | \_interestRate |

Define the following method in the **SavingAccount**class.

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public void Display() | This method is used to call the base class Display() and in addition displays \_interestRate. |

Include **default** and **parameterized** constructor for the class.  
Prototype for the Parameterized Constructor **SavingAccount(string \_holderName, long \_accountNumber, string \_IFSCCode, long \_contactNumber, double \_interestRate)**  
Use **base**Keyword to call the base class constructor.

Consider the class **CurrentAccount**which inherits **Account** class with the following private attributes/variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| string | \_organizationName |
| long | \_TIN |

Define the following method in the **CurrentAccount**class

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public void Display() | This method is used to call base class Display() and in addition displays \_organizationName,\_TIN |

Include **default** and **parameterized** constructor for the class.  
Prototype for the Parameterized Constructor **CurrentAccount(string \_holderName, long \_accountNumber, string \_IFSCCode, long \_contactNumber,string \_organizationName, long \_TIN)**  
Use **base**Keyword to call the base class constructor.

Create **Program** 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)

**Steffan,982714210,S160030600514,9092304676**

Enter Account Type

**saving**

Enter Interest Rate

**12.0**

Your Contact Details

HolderName : Steffan

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)

**John,7889142075,S1600ABY0576,9944001700**

Enter Account Type

**current**

Enter organization Name

**pentamedia Graphics Limited**

Enter TIN number

**7841**

Your Contact Details

HolderName : John

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)  
**Shira,987451024,SWQ78914AF,9078425168**  
Enter Account Type  
**curr**  
Enter valid Account Type

**GST Calculation**

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%.

**[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.]**  
  
Consider a class names **Event** with the following protected attributes.

|  |  |
| --- | --- |
| **Data Type** | **Attributes** |
| string | \_name |
| string | \_type |
| double | \_costPerDay |
| int | \_noOfDays |

Include **default** and **parameterized** constructor for the class.  
Prototype for the Parameterized Constructor **Event(string \_name, string \_type, double \_costPerDay, int \_noOfDays)**  
  
Consider the class **Exhibition** which inherits the **Event** class with the following private attributes.

|  |  |
| --- | --- |
| **Data Type** | **Attributes** |
| static int | \_gst = 5 |
| int | \_noOfStalls |

Include **default** and **parameterized** constructor for the class.  
Prototype for the Parameterized Constructor **Exhibition(string \_name, string \_type, double \_costPerDay, int \_noOfDays, int \_noOfStalls)**

Define the following method in the **Exhibition** class.

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

Consider the class **StageEvent** which inherits the **Event** class with the following private attributes.

|  |  |
| --- | --- |
| **Data Type** | **Attributes** |
| static int | \_gst = 15 |
| int | \_noOfSeats |

Include **default** and **parameterized** constructor for the class.  
Prototype for the Parameterized Constructor **StageEvent(string \_name, string \_type, double \_costPerDay, int \_noOfDays, int \_noOfSeats)**  
  
Define the following method in the **StageEvent** class.

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

Use base() to call and assign values in base class constructor.  
Override ToString() method to display the event details and the total amount inside this ToString() method.  
  
Create **Program** 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. Display the statement ‘**Event Details**’ inside the Main() method.

**Input and Output Format:**

All the double values should be displayed upto **2 decimal** palces

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