**1.Constructors in Inheritance -  Super**

As always, a sudden thought comes to our mind. We do save lot of time by reusing the members and functions written in the parent class. Does the same apply to constructors? You quickly learn there is a super keyword with which you can pass parameters and invoke a constructor of the parent class. Common !  Lets translate it into code.  
 Create a class **Customer** with following private data members

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

 Methods in class **Customer.**

|  |  |
| --- | --- |
| **Method Name** | **Function** |
| displayCustomer() | This method displays the details of the customer with the total bill amount and  discount amount under privilege customer and  senior citizen customer |

Create a class **SeniorCitizenCustomer** which extends the class **Customer**.  
Use **super** keyword to invoke parent class constructor.

Methods in class **SeniorCitizenCustomer**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Function** | **Return Type** |
|  | Include a 4 arguments constructor with the appropriate arguments.The order in which the arguements should be passed is name, address, age, mobileNumber. |  |
| getBillAmount(amount) | To calculate the payment amount where discount is 12% of amount | Double |

Create the class **PrivilegeCustomer** which extends the class **Customer** .  
Use **super** keyword to invoke parent class constructor.

Methods in class **PrivilegeCustomer**

|  |  |  |
| --- | --- | --- |
| **Method Name** | **Function** | **Return Type** |
|  | Include a 4 arguments constructor with the appropriate arguments.The order in which the arguements should be passed is name,address, age, mobileNumber. |  |
| getBillAmount(amount) | To calculate the payment amount where discount is 30% of amount | Double |

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

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

**Sample Input and Output 1:**

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

**Sample Input and Output 2:**  
1)Privilege Customer  
2)SeniorCitizen Customer  
Enter Customer Type  
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

**2,More Inhertiance - Mutilevel Inheritance**

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

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

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

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

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

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

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

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

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

Methods in class **RewardsCreditCard**

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

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

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

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

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

Use Appropriate Getters Setters for the above classes.

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

**Sample Input and Output 1:**

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

Enter the travel details

Travel Place

**Banglore**

Number of tickets

**2**

Cost per ticket

**1500**

1)Travel Creditcard

2)RewardCreditcard

Enter credit card type

**1**

1)International

2)National

Enter travel creditcard type

**1**

Enter the creditcard number

**123456794**

Enter the creditcard holdername

**Praveen**

Enter the available amount

**65255**

Hello Praveen, You have to pay Rs2700.0

**Sample Input and Output 2:**

Enter the travel details

Travel Place

**Chennai**

Number of tickets

**20**

Cost per ticket

**150**

1)Travel Creditcard

2)RewardCreditcard

Enter credit card type

**3**

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

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

**3.Discounts - Inheritance**

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

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

Methods in class **Customer**

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

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

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

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

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

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

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

**Note :**

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

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

**Input and Output Format:**

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

**Sample Input and Output 2:**

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

**4.Parking Lot - Requirement**

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

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

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

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

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

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

The Vehicle and Ticket details should be given as a comma-separated value in the below order,  
registrationNo, name, type, weight,ticketNo,parkedTime,cost  
When the “vehicle” object is printed, it should display the following details  
Print format:  
**System.out.format(**"%-15s %-10s %-12s %-7s %s\n","Registration No","Name","Type","Weight","Ticket No"**);**  
**Note:**The vehicle lists are displayed in the main method.  
            If any other choice is selected, display "**Invalid Choice**"  
            If the search detail is not found, display "No such vehicle is present"  
            Display one digit after the decimal point for Double Datatype.  
  
**Sample test case 1:**  
Enter the number of vehicles:

6

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

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

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

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

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

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

Enter a search type:

1.By type

2.By parked time

**1**

Enter the vehicle type

**FourWheeler**

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

**Sample test case 2:**

Enter the number of vehicles:

**6**

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

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

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

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

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

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

Enter a search type:

1.By type

2.By parked time

**2**

Enter the parked time:

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

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

**5.Abstract class and methods**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Create a driver class called  **Main** . In the Main method, obtain input from the user in CSV format and split the input using split method.  
  
**Note:** The products or packages in instasmart are rented based on the month, hence the total cost of the package has to be divided by the number of months(rental period) requested by the customer.  
  
**Input and Output format:**  
Refer sample Input and Output for formatting specifications.  
   
**[All text in bold corresponds to input and the rest corresponds to output]**

**Sample Input and Output 1:**  
1.Custom Package  
2.Living Room Package  
Enter your choice  
**1**  
Enter the custom package details in comma separated format:  
**Package-1,1,Chennai,4,1500.55,Steel,3**  
Custom Package details:  
Package name : Package-1  
Availability : true  
City : Chennai  
Rental period : 4  
Product type : Steel  
No. of products : 3  
Total Cost : 1500.55  
Cost of package/month : 375.14  
  
**Sample Input and Output 2:**  
1.Custom Package  
2.Living Room Package  
Enter your choice  
**2**  
Enter the living room package details in comma separated format:  
**Package-1,0,Chennai,3,1500.55,4,2,3**  
Living Room Package details:  
Package name : Package-1  
Availability : false  
City : Chennai  
Rental period : 3  
No. of Fan : 4  
No. of Chair : 2  
No. of Wardrobe : 3  
Total Cost : 1500.55  
Cost of package/month : 500.18  
  
**Sample Input and Output 3:**  
1.Custom Package  
2.Living Room Package  
Enter your choice  
**3**  
Invalid choice

**6.Polymorphism with Interface**

Write a Java program to display the final balance amount of an account after a transaction using interface and polymorphism.**Strictly adhere to the Object Oriented Specifications given in the problem statement. All class names, member variable names and function names should be the same as specified in the problem statement.**

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

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

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

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

**Note:**aadharId in India is similar to SSN in USA  
Include appropriate **getters**and **setters**and **constructors**for the above class.  
Implement the following overloaded abstract methods in **Payee**class.

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

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

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

**Sample Input and Output 1:**

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

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

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

**7.Search Customer**

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

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

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

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

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

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

**Note:**

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

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

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

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

**Sample Input and Output 1:**

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

**Sample Input and Output 2:**

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

**Sample Input and Output 3:**

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

**8.Hall management**

**[Note :  
Strictly adhere to the object oriented specifications given as a part of the problem statement.  
Follow the naming conventions as mentioned. Create separate classes in separate files.]**  
 Create a class named **Hall**with the following private attributes/variables.

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

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

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

Create a **Main** class with Main method, to test the above class.  
Get the hall details from the user the perform add and remove operation on list.  
  
**Note:**  
Display the hall details in the following format given in sample output.  
**Input format:**  
The first line of input is an integer which corresponds to the choice of which action to be done.  
The input to add the hall details is in the CSV format [**Hall name, Contact number, Cost per day, Owner name**].  
The input to remove the hall details is an integer that corresponds to the hall index number. The index starts from 0.  
  
**Input and Output Format**  
Refer sample input and output for formatting specifications.  
**All text in bold corresponds to the input and the rest corresponds to output.**

**Sample Input and Output 1:**  
Menu  
1.Add Hall  
2.Remove Hall  
Enter your choice  
**1**  
Enter the Hall details in CSV format  
**Party hall,9876543210,4000.0,Jarviz**  
Hall created successfully  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Hall  
2.Remove Hall  
Enter your choice  
**1**  
Enter the Hall details in CSV format  
**Disco hall,9876543201,5000.0,Starc**  
Hall created successfully  
Do you want to continue(y/n)?:  
**y**  
Menu  
1.Add Hall  
2.Remove Hall  
Enter your choice  
**2**  
Enter the index of the hall to be removed  
**0**  
Hall removed successfully  
Do you want to continue(y/n)?:  
**n**  
Hall Details  
Name           ContactNumber  CostPerDay     OwnerName  
Disco hall     9876543201     5000.0           Starc  
  
**Sample Input and Output 2:**

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

**Sample Input and Output 3:**

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

**9.Customer - SubList**

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

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

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

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

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

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

**Note:**

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

**Input and Output Format:**

Refer sample input and output for formatting specifications.

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

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