public static List<User> prefill(){

List<User> userList=new ArrayList<>();

userList.add(new User("Harry","harry@gmail.com","9856231478","NewYork", new ArrayList<Purchase>()));

userList.add(new User("Joe","joe@gmail.com","8956231548","Chicago", new ArrayList<Purchase>()));

userList.add(new User("Oliver","oliver@gmail.com","7856124589","SanFrancisco", new ArrayList<Purchase>()));

userList.add(new User("Danny","danny@gmail.com","9382714568","Los Angeles", new ArrayList<Purchase>()));

userList.add(new User("Brandon","brandon@gmail.com","98596215488","Boston", new ArrayList<Purchase>()));

userList.add(new User("Rob","rob@gmail.com","9584596662","NewYork", new ArrayList<Purchase>()));

return userList;

}

public static List<Item> prefill(){

List<Item> list = new ArrayList<>();

list.add(new Item("Paneer Fried Rice",Double.parseDouble("150"),"Veg"));

list.add(new Item("Chicken Fried Rice",Double.parseDouble("210"),"NonVeg"));

list.add(new Item("Bucket Chicken",Double.parseDouble("479"),"Veg"));

list.add(new Item("Ghee Roast",Double.parseDouble("75"),"Veg"));

list.add(new Item("Masal Roast",Double.parseDouble("65"),"Veg"));

list.add(new Item("Grill",Double.parseDouble("250"),"NonVeg"));

list.add(new Item("Tomato Soup",Double.parseDouble("100"),"Veg"));

list.add(new Item("Prawn Fry",Double.parseDouble("185"),"NonVeg"));

list.add(new Item("Mushroom Briyani",Double.parseDouble("140"),"Veg"));

list.add(new Item("Baby Corn Chilli",Double.parseDouble("80"),"Veg"));

list.add(new Item("Barbeque Chicken",Double.parseDouble("400"),"NonVeg"));

list.add(new Item("Veg Extravaganza",Double.parseDouble("99"),"Veg"));

list.add(new Item("Margherita",Double.parseDouble("399"),"Veg"));

list.add(new Item("Tandoori Chicken",Double.parseDouble("500"),"NonVeg"));

list.add(new Item("Mutton Balls",Double.parseDouble("90"),"NonVeg"));

return list;

}

}​

**Food aggregator - Requirement 1**

You being an aspiring entrepreneur wish to start a Food aggregator app of your own. You have collected the details of menu and user details from hotels around your city. Now it's time to develop your app by creating domains and integrate collected details into it. You have chosen four domains for your app namely Item, Order, Purchase and User. Let's add features to your app one by one.  
  
**Requirement 1:**

Let’s start off by creating two **User** objects and check whether they are equal.

1. Create a **User** Class with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| name | String |
| email | String |
| phoneNumber | String |
| location | String |

1. Mark all the attributes as private
2. Create / Generate appropriate Getters & Setters
3. Add a default constructor and a parameterized constructor to take in all attributes in the given order:   
   **User ( String name, String email, String phoneNumber , String location )**
4. When the “User” object is printed, it should display the following details: **[Override the toString method]**  
   Print format:  
   Name: "name"  
   Email: "email"  
   PhoneNumber: "phoneNumber "  
   Location: "location"
5. Two Users are considered the same if they have the same name, and email. Implement the logic in the appropriate function. (Case – Insensitive) **[Override the equals method]**  
     
   The input format consists of User details separated by a comma in the below order,  
   **name, email, phoneNumber , location**

The Input to your program would be details of two Users, you need to display their details as given in "5th point(refer above)" and compare the two Users and display if the Users are same or different.  
  
**Note:**There is an empty line between display statements. Print the empty lines in the main method.  
  
**Sample Input/Output:**

Enter user 1 detail:

**Oliver,oliver@gmail.com,7856124589,SanFrancisco**

Enter user 2 detail:

**Harry,harry@gmail.com,9856231478,NewYork**

User 1

Name: Oliver

Email: oliver@gmail.com

PhoneNumber: 7856124589

Location: SanFrancisco

User 2

Name: Harry

Email: harry@gmail.com

PhoneNumber: 9856231478

Location: NewYork

User 1 and User 2 are different  
  
**Sample Input/Output 2:**

Enter user 1 detail:

**Harry,harry@gmail.com,9856231478,NewYork**

Enter user 2 detail:

**haRRy,harry@gmail.com,9856231478** **,NewYork**

User 1

Name: Harry

Email: harry@gmail.com

PhoneNumber: 9856231478

Location: NewYork

User 2

Name: haRRy

Email: harry@gmail.com

PhoneNumber: 9856231478

Location: NewYork

User 1 is same as User 2

**Test Cases**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [Assign Groups](https://cognizant.e-box.co.in/problem/showTestcaseGroup/8559)   | **SNo** | **Name** | **Input** | **Output** | **Purpose** | **Sample** | | --- | --- | --- | --- | --- | --- | | 1 | t4 | Sam,sam@gmail.com,8525852585,Boston Dean,dean@gmail.com,7845962354,Los Angeles | Enter user 1 detail: Enter user 2 detail:  User 1 Name: Sam Email: sam@gmail.com PhoneNumber: 8525852585 Location: Boston  User 2 Name: Dean Email: dean@gmail.com PhoneNumber: 7845962354 Location: Los Angeles  User 1 and User 2 are different | General |  | | 2 | t5 | James,james@gmail.com,9856321475,Chicago jaMeS,james@gmail.com,9856321475,Chicago | Enter user 1 detail: Enter user 2 detail:  User 1 Name: James Email: james@gmail.com PhoneNumber: 9856321475 Location: Chicago  User 2 Name: jaMeS Email: james@gmail.com PhoneNumber: 9856321475 Location: Chicago  User 1 is same as User 2 | General |  | | 3 | t3 | Harry,harry@gmail.com,9856231478,NewYork harry,harry@gmail.com,8525852585,Boston | Enter user 1 detail: Enter user 2 detail:  User 1 Name: Harry Email: harry@gmail.com PhoneNumber: 9856231478 Location: NewYork  User 2 Name: harry Email: harry@gmail.com PhoneNumber: 8525852585 Location: Boston  User 1 is same as User 2 | General |  | | 4 | t2 | Harry,harry@gmail.com,9856231478,NewYork haRRy,harry@gmail.com,9856231478,NewYork | Enter user 1 detail: Enter user 2 detail:  User 1 Name: Harry Email: harry@gmail.com PhoneNumber: 9856231478 Location: NewYork  User 2 Name: haRRy Email: harry@gmail.com PhoneNumber: 9856231478 Location: NewYork  User 1 is same as User 2 | General |  | | 5 | t1 | Oliver,oliver@gmail.com,7856124589,SanFrancisco Harry,harry@gmail.com,9856231478,NewYork | Enter user 1 detail: Enter user 2 detail:  User 1 Name: Oliver Email: oliver@gmail.com PhoneNumber: 7856124589 Location: SanFrancisco  User 2 Name: Harry Email: harry@gmail.com PhoneNumber: 9856231478 Location: NewYork  User 1 and User 2 are different | General |  | |

**Food Aggregator - Requirement 2**

**Requirement 2:**  
   In this requirement, you need to validate the email of the User.  
  
a)Create a class **Main** with the following static methods:

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| static Boolean validateEmail(String email) | Validate the Email based on the rules given below. Returns **true**ifEmail is valid else return **false** |

b) While validating email follow the below rules. The format of the email id is given below  
   
            **username@domain.TLD**  
  
where, TLD - Top Level Domain  
  
1. The email should start only with alphabets(lowercase).  
2. The username part of email can contain alphabets(lowercase), numbers and the special characters (  .   and   \_    ).  
3. The username part of the email should not contain any special characters other than**" . "**and**" \_ "**.  
4. After the username special character @ should present.  
5. The email domain should contain only alphabets(lowercase).  
6. After the email domain, a dot ( **.** ) should be present.  
7. The email Top Level Domain should contain only alphabets(lowercase) and it should have only 2 to 6 characters.  
  
Example: **harson\_Wells.01@google.com** is a valid email id.  
  
**Note:** Print "**Email is valid**" if email is valid else print "**Email is invalid**".  
            All the above print statements are present in the main method.  
  
**[All text in bold corresponds to input]**  
**Sample Input and Output 1:**  
  
Enter the email to be validated:  
**harry@gmail.com**  
Email is valid  
  
**Sample Input and Output 2:**  
  
Enter the email to be validated:  
**jerin$80@yahoo.com**  
Email is invalid

**Test Cases**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [Assign Groups](https://cognizant.e-box.co.in/problem/showTestcaseGroup/8556)   | **SNo** | **Name** | **Input** | **Output** | **Purpose** | **Sample** | | --- | --- | --- | --- | --- | --- | | 1 | 1 | harry@gmail.com | Enter the email to be validated: Email is valid | General |  | | 2 | 5 | harish@gmail.unitedstates | Enter the email to be validated: Email is invalid | General |  | | 3 | 4 | hendry\_96@gmail.com | Enter the email to be validated: Email is valid | General |  | | 4 | 10 | harry@hotmail.com9 | Enter the email to be validated: Email is invalid | General |  | | 5 | 9 | harry.s@hotmail90.com | Enter the email to be validated: Email is invalid | General |  | | 6 | 11 | harry@hotmail.c | Enter the email to be validated: Email is invalid | General |  | | 7 | 8 | yuvan.s@hotmail.com | Enter the email to be validated: Email is valid | General |  | | 8 | 6 | \_harrypotter@hotmail.com | Enter the email to be validated: Email is invalid | General |  | | 9 | 3 | andrew45@yahoo.com | Enter the email to be validated: Email is valid | General |  | | 10 | 7 | harishjeyaraj+a@gmail.com | Enter the email to be validated: Email is invalid | General |  | | 11 | 2 | jerin$80@yahoo.com | Enter the email to be validated: Email is invalid | General |  | |

**Food aggregator - Requirement 3**

**Requirement 3:**  
In this requirement develop a feature in which you can search a List of Items by type, and price.  
  
a) Create a class **Item** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| name | String |
| price | Double |
| type | String |

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:  
**Item( String name, Double price, String type)**  
  
b) Create a class **ItemBO**with the following methods,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public List<Item> findItem(List<Item> itemList, String type) | This method accepts a list of items and type as arguments and returns a list of items that match with the given type. |
| public List<Item> findItem(List<Item> itemList, Double price) | This method accepts a list of items and price as arguments and returns a list of items that match with the given price. |

The item details should be given as a comma-separated value in the below order,  
**name, price, type**  
  
Print format:  
**System.out.format("%-20s %-5s %s\n","Name","Price","Type");**  
  
**Note:**The item 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 item is present"  
            Display one digit after the decimal point for Double Datatype.  
  
**Sample Input and Output 1:**  
Enter the number of items:  
**4  
Paneer Fried Rice,150,Veg  
Chicken Fried Rice,210,NonVeg  
Bucket Chicken,479,NonVeg  
Ghee Roast,75,Veg**  
Enter a search type:  
1.By Type  
2.By Price  
**1**  
Enter the Type:  
**Veg**

|  |  |  |
| --- | --- | --- |
| Name | Price | Type |
| Paneer Fried Rice | 150.0 | Veg |
| Ghee Roast | 75.0 | Veg |

**Sample Input and Output 2:**  
Enter the number of items:  
**4  
Paneer Fried Rice,150,Veg  
Chicken Fried Rice,210,NonVeg  
Bucket Chicken,479,NonVeg  
Ghee Roast,75,Veg**  
Enter a search type:  
1.By Type  
2.By Price  
**1**  
Enter the type:  
**Meals**  
No such item is present

**Sample Input and Output 3:**  
Enter the number of items:  
**4  
Paneer Fried Rice,150,Veg  
Chicken Fried Rice,210,NonVeg  
Bucket Chicken,479,NonVeg  
Ghee Roast,75,Veg**  
Enter a search type:  
1.By Type  
2.By Price  
**3**  
Invalid choice

**Test Cases**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [Assign Groups](https://cognizant.e-box.co.in/problem/showTestcaseGroup/8558)   | **SNo** | **Name** | **Input** | **Output** | **Purpose** | **Sample** | | --- | --- | --- | --- | --- | --- | | 1 | TC6 | 6 Baby Corn Chill,80,Veg Barbeque Chicken,400,NonVeg Veg Extravaganza,399,Veg Margherita,399,Veg Tandoori Chicken,500,NonVeg Mutton Balls,90,NonVeg 2 399 | Enter the number of items: Enter a search type: 1.By Type 2.By Price Enter the Price: Name                 Price Type Veg Extravaganza     399.0 Veg Margherita           399.0 Veg | General |  | | 2 | TC2 | 4 Paneer Fried Rice,150,Veg Chicken Fried Rice,210,NonVeg Bucket Chicken,479,NonVeg Ghee Roast,75,Veg 2 479 | Enter the number of items: Enter a search type: 1.By Type 2.By Price Enter the Price: Name                 Price Type Bucket Chicken       479.0 NonVeg | General |  | | 3 | TC7 | 6 Baby Corn Chill,80,Veg Barbeque Chicken,400,NonVeg Veg Extravaganza,399,Veg Margherita,399,Veg Tandoori Chicken,500,NonVeg Mutton Balls,90,NonVeg -1 | Enter the number of items: Enter a search type: 1.By Type 2.By Price Invalid choice | General |  | | 4 | TC1 | 4 Paneer Fried Rice,150,Veg Chicken Fried Rice,210,NonVeg Bucket Chicken,479,NonVeg Ghee Roast,75,Veg 1 Veg | Enter the number of items: Enter a search type: 1.By Type 2.By Price Enter the Type: Name                 Price Type Paneer Fried Rice    150.0 Veg Ghee Roast           75.0  Veg | General |  | | 5 | TC5 | 6 Baby Corn Chill,80,Veg Barbeque Chicken,400,NonVeg Veg Extravaganza,399,Veg Margherita,399,Veg Tandoori Chicken,500,NonVeg Mutton Balls,90,NonVeg 1 NonVeg | Enter the number of items: Enter a search type: 1.By Type 2.By Price Enter the Type: Name                 Price Type Barbeque Chicken     400.0 NonVeg Tandoori Chicken     500.0 NonVeg Mutton Balls         90.0  NonVeg | General |  | | 6 | TC8 | 6 Baby Corn Chill,80,Veg Barbeque Chicken,400,NonVeg Veg Extravaganza,399,Veg Margherita,399,Veg Tandoori Chicken,500,NonVeg Mutton Balls,90,NonVeg 2 1100 | Enter the number of items: Enter a search type: 1.By Type 2.By Price Enter the Price: No such item is present | General |  | | 7 | TC4 | 4 Paneer Fried Rice,150,Veg Chicken Fried Rice,210,NonVeg Bucket Chicken,479,NonVeg Ghee Roast,75,Veg 1 Meals | Enter the number of items: Enter a search type: 1.By Type 2.By Price Enter the Type: No such item is present | General |  | | 8 | TC3 | 4 Paneer Fried Rice,150,Veg Chicken Fried Rice,210,NonVeg Bucket Chicken,479,NonVeg Ghee Roast,75,Veg 3 | Enter the number of items: Enter a search type: 1.By Type 2.By Price Invalid choice | General |  | |

**Food aggregator - Requirement 4**

**Requirement 4:**  
  
In this requirement, you need to sort the list of items based on name and price.  
  
a) Create a class **Item** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| name | String |
| price | Double |
| type | String |

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: **Item( String name, Double  price, String type )**  
  
b) Create the following static methods in the Item class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| static Item createItem(String detail) | This method accepts a String. The item detail separated by commas is passed as the argument. Split the details and create an item object and returns it. |

The item details should be given as a comma-separated value in the below order,  
name, type, price  
  
c) The Item class should implement the **Comparable** interface which sorts the Item list based on **name**(case-insensitive). While comparing, all the name attributes in the list are unique.  
  
d) Create a class **PriceComparator** which implements Comparator interface and sort the Item list based on **price**. While comparing, all the price attributes in the list are unique.  
  
Get the number of Items and item details and create an item list. Sort the Items according to the given option and display the list.  
  
When the “item” object is printed, it should display the following details  
Print format:  
**System.out.format("%-20s %-10s %-12s\n","Name","Price","Type");**  
**Note:** Display one digit after decimal point for Double datatype.   
  
**Sample Input & Output 1:**   
  
Enter the number of items:   
**4  
Paneer Fried Rice,150.0,Veg  
Chicken Fried Rice,210.0,NonVeg  
Bucket Chicken,479.0,NonVeg  
Ghee Roast,75.0,Veg**   
Enter a type to sort:   
1.Sort by Name   
2.Sort by Price   
**1**   
Name                              Price      Type           
Bucket Chicken         479.0      NonVeg         
Chicken Fried Rice   210.0      NonVeg         
Ghee Roast                  75.0        Veg            
Paneer Fried Rice     150.0      Veg    
  
  
**Sample Input & Output 2:**   
  
Enter the number of items:   
**4  
Paneer Fried Rice,150.0,Veg  
Chicken Fried Rice,210.0,NonVeg  
Bucket Chicken,479.0,NonVeg  
Ghee Roast,75.0,Veg**   
Enter a type to sort:   
1.Sort by Name   
2.Sort by Price   
**2**   
Name                              Price      Type           
Ghee Roast                  75.0        Veg            
Paneer Fried Rice     150.0      Veg            
Chicken Fried Rice   210.0      NonVeg         
Bucket Chicken         479.0      NonVeg

**Test Cases**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [Assign Groups](https://cognizant.e-box.co.in/problem/showTestcaseGroup/8557)   | **SNo** | **Name** | **Input** | **Output** | **Purpose** | **Sample** | | --- | --- | --- | --- | --- | --- | | 1 | 4 | 8 Masal Roast,65.0,Veg Grill,250.0,NonVeg Tomato Soup,100.0,Veg Prawn Fry,185.0,NonVeg Mushroom Briyani,140.0,Veg Baby Corn Chill,80.0,Veg Barbeque Chicken,400.0,NonVeg Veg Extravaganza,299.0,Veg 2 | Enter the number of items: Enter a type to sort: 1.Sort by Name 2.Sort by Price Name                 Price      Type         Masal Roast          65.0       Veg          Baby Corn Chill      80.0       Veg          Tomato Soup          100.0      Veg          Mushroom Briyani     140.0      Veg          Prawn Fry            185.0      NonVeg       Grill                250.0      NonVeg       Veg Extravaganza     299.0      Veg          Barbeque Chicken     400.0      NonVeg | General |  | | 2 | 1 | 4 Paneer Fried Rice,150.0,Veg Chicken Fried Rice,210.0,NonVeg Bucket Chicken,479.0,NonVeg Ghee Roast,75.0,Veg 1 | Enter the number of items: Enter a type to sort: 1.Sort by Name 2.Sort by Price Name                 Price      Type         Bucket Chicken       479.0      NonVeg       Chicken Fried Rice   210.0      NonVeg       Ghee Roast           75.0       Veg          Paneer Fried Rice    150.0      Veg | General |  | | 3 | 5 | 15 Paneer Fried Rice,150.0,Veg Chicken Fried Rice,210.0,NonVeg Bucket Chicken,479.0,NonVeg Ghee Roast,75.0,Veg Masal Roast,65.0,Veg Grill,250.0,NonVeg Tomato Soup,100.0,Veg Prawn Fry,185.0,NonVeg Mushroom Briyani,140.0,Veg Baby Corn Chill,80.0,Veg Barbeque Chicken,400.0,NonVeg Veg Extravaganza,299.0,Veg Margherita,399.0,Veg Tandoori Chicken,500.0,NonVeg Mutton Balls,90.0,NonVeg 1 | Enter the number of items: Enter a type to sort: 1.Sort by Name 2.Sort by Price Name                 Price      Type         Baby Corn Chill      80.0       Veg          Barbeque Chicken     400.0      NonVeg       Bucket Chicken       479.0      NonVeg       Chicken Fried Rice   210.0      NonVeg       Ghee Roast           75.0       Veg          Grill                250.0      NonVeg       Margherita           399.0      Veg          Masal Roast          65.0       Veg          Mushroom Briyani     140.0      Veg          Mutton Balls         90.0       NonVeg       Paneer Fried Rice    150.0      Veg          Prawn Fry            185.0      NonVeg       Tandoori Chicken     500.0      NonVeg       Tomato Soup          100.0      Veg          Veg Extravaganza     299.0      Veg | General |  | | 4 | 7 | 4 PANEER Fried Rice,150.0,Veg pepper mint,110.0,NonVeg Pepper Omlet,20.0,veg paneeR soda,25.0,veg 1 | Enter the number of items: Enter a type to sort: 1.Sort by Name 2.Sort by Price Name                 Price      Type         PANEER Fried Rice    150.0      Veg          paneeR soda          25.0       veg          pepper mint          110.0      NonVeg       Pepper Omlet         20.0       veg | General |  | | 5 | 6 | 15 Paneer Fried Rice,150.0,Veg Chicken Fried Rice,210.0,NonVeg Bucket Chicken,479.0,NonVeg Ghee Roast,75.0,Veg Masal Roast,65.0,Veg Grill,250.0,NonVeg Tomato Soup,100.0,Veg Prawn Fry,185.0,NonVeg Mushroom Briyani,140.0,Veg Baby Corn Chill,80.0,Veg Barbeque Chicken,400.0,NonVeg Veg Extravaganza,299.0,Veg Margherita,399.0,Veg Tandoori Chicken,500.0,NonVeg Mutton Balls,90.0,NonVeg 2 | Enter the number of items: Enter a type to sort: 1.Sort by Name 2.Sort by Price Name                 Price      Type         Masal Roast          65.0       Veg          Ghee Roast           75.0       Veg          Baby Corn Chill      80.0       Veg          Mutton Balls         90.0       NonVeg       Tomato Soup          100.0      Veg          Mushroom Briyani     140.0      Veg          Paneer Fried Rice    150.0      Veg          Prawn Fry            185.0      NonVeg       Chicken Fried Rice   210.0      NonVeg       Grill                250.0      NonVeg       Veg Extravaganza     299.0      Veg          Margherita           399.0      Veg          Barbeque Chicken     400.0      NonVeg       Bucket Chicken       479.0      NonVeg       Tandoori Chicken     500.0      NonVeg | General |  | | 6 | 2 | 4 Paneer Fried Rice,150.0,Veg Chicken Fried Rice,210.0,NonVeg Bucket Chicken,479.0,NonVeg Ghee Roast,75.0,Veg 2 | Enter the number of items: Enter a type to sort: 1.Sort by Name 2.Sort by Price Name                 Price      Type         Ghee Roast           75.0       Veg          Paneer Fried Rice    150.0      Veg          Chicken Fried Rice   210.0      NonVeg       Bucket Chicken       479.0      NonVeg | General |  | | 7 | 3 | 8 Masal Roast,65.0,Veg Grill,250.0,NonVeg Tomato Soup,100.0,Veg Prawn Fry,185.0,NonVeg Mushroom Briyani,140.0,Veg Baby Corn Chill,80.0,Veg Barbeque Chicken,400.0,NonVeg Veg Extravaganza,299.0,Veg 1 | Enter the number of items: Enter a type to sort: 1.Sort by Name 2.Sort by Price Name                 Price      Type         Baby Corn Chill      80.0       Veg          Barbeque Chicken     400.0      NonVeg       Grill                250.0      NonVeg       Masal Roast          65.0       Veg          Mushroom Briyani     140.0      Veg          Prawn Fry            185.0      NonVeg       Tomato Soup          100.0      Veg          Veg Extravaganza     299.0      Veg | General |  | |

**Food Aggregator Requirement - 5**

**Requirement 5:**  
In this requirement develop a feature to compute the price for every purchase based on the price of each item, quantity and coupon code.  
  
a) Create a class **Purchase** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| id | Integer |
| price | Double |
| couponCode | String |
| purchaseDate | java.util.Date |
| orderList | List<Order> |

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:  
**Purchase( Integer id, String couponCode, Date purchaseDate, List<Order> orderList)**  
  
b) Create the following methods for **Purchase**class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public static void computePrice(List<Purchase> list) | This method accepts a list of purchase objects as arguments and sets the computed price value for each purchase. |

c) Create a class **Order** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| quantity | Integer |
| item | Item |

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:  
**Order( Integer quantity, Item item)**  
  
d) Create a class **Item** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| name | String |
| price | Double |
| type | String |

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:  
**Item( String name, String type, Double price )**  
  
e) Create the following methods for **Item**class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public static List<Item> prefill() | This method returns a list of prefilled Item objects |

The purchase details should be given as a comma-separated value in the below order,  
**id, couponCode** **, purchaseDate**  
The order details should be given as a comma-separated value in the below order,  
**quantity, itemName**   
  
Print format:  
**System.out.format("%-5s %-10s %-12s %s\n","Id","Price","Coupon Code","Purchase Date");  
  
Note:**use the prefill() method, given in the template, to get the Item objects.  
               Display one digit after the decimal point for Double Datatype.   
  
While computing price, multiply the quantity with the item's price in the orderList.  
If anyone of the coupon codes is applied, discount the price based on the below criteria.

|  |  |
| --- | --- |
| FIRST | Rs.100 discount |
| BUYFIVE | Rs.500 discount |
| ORDER50 | 50% discount |
| ORDER75 | 75% discount |
| DEAL25 | 25% discount |
| CHICKEN70 | 70% discount |

**Sample Input and Output:**  
  
Enter the number of Purchase  
**2**  
Enter purchase detail 1  
**1,FIRST,12-02-2018**  
Enter the number of Orders  
**2  
2,Grill  
1,Margherita**  
Enter purchase detail 2  
**2,ORDER50,13-02-2018**  
Enter the number of Orders  
**3  
2,Tandoori Chicken  
1,Barbeque Chicken  
1,Ghee Roast**  
Id    Price      Coupon Code  Purchased Date  
1     799.0      FIRST                 12-02-2018  
2     737.5      ORDER50        13-02-2018

**Test Cases**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [Assign Groups](https://cognizant.e-box.co.in/problem/showTestcaseGroup/8560)   | **SNo** | **Name** | **Input** | **Output** | **Purpose** | **Sample** | | --- | --- | --- | --- | --- | --- | | 1 | TC1 | 2 1,FIRST,12-02-2018 2 2,Grill 1,Margherita 2,ORDER50,13-02-2018 3 2,Tandoori Chicken 1,Barbeque Chicken 1,Ghee Roast | Enter the number of Purchases Enter purchase detail 1 Enter the number of Orders Enter purchase detail 2 Enter the number of Orders Id    Price      Coupon Code  Purchase Date 1     799.0      FIRST        12-02-2018 2     737.5      ORDER50      13-02-2018 | General |  | | 2 | TC3 | 4 4,ORDER75,16-05-2018 1 3,Baby Corn Chilli 3,ORDER50,04-06-2018 2 1,Tandoori Chicken 1,Prawn Fry 2,FIRST,15-05-2018 1 1,Bucket Chicken 5,DEAL25,28-04-2018 2 2,Margherita 1,Barbeque Chicken | Enter the number of Purchases Enter purchase detail 1 Enter the number of Orders Enter purchase detail 2 Enter the number of Orders Enter purchase detail 3 Enter the number of Orders Enter purchase detail 4 Enter the number of Orders Id    Price      Coupon Code  Purchase Date 4     60.0       ORDER75      16-05-2018 3     342.5      ORDER50      04-06-2018 2     379.0      FIRST        15-05-2018 5     898.5      DEAL25       28-04-2018 | General |  | | 3 | TC4 | 4 3,ORDER50,04-06-2018 1 1,Prawn Fry 6,CHICKEN70,10-04-2018 1 2,Tandoori Chicken 1,FIRST,01-03-2018 3 1,Veg Extravaganza 1,Margherita 1,Paneer Fried Rice 2,ORDER50,04-06-2018 5 1,Chicken Fried Rice 1,Bucket Chicken 1,Grill 1,Baby Corn Chilli 1,Prawn Fry | Enter the number of Purchases Enter purchase detail 1 Enter the number of Orders Enter purchase detail 2 Enter the number of Orders Enter purchase detail 3 Enter the number of Orders Enter purchase detail 4 Enter the number of Orders Id    Price      Coupon Code  Purchase Date 3     92.5       ORDER50      04-06-2018 6     300.0      CHICKEN70    10-04-2018 1     548.0      FIRST        01-03-2018 2     602.0      ORDER50      04-06-2018 | General |  | | 4 | TC2 | 3 1,DEAL25,28-04-2018 2 1,Grill 2,Tomato Soup 2,BUYFIVE,15-05-2018 3 2,Margherita 1,Mutton Balls 2,Chicken Fried Rice 4,ORDER20,26-03-2018 1 5,Masal Roast | Enter the number of Purchases Enter purchase detail 1 Enter the number of Orders Enter purchase detail 2 Enter the number of Orders Enter purchase detail 3 Enter the number of Orders Id    Price      Coupon Code  Purchase Date 1     337.5      DEAL25       28-04-2018 2     808.0      BUYFIVE      15-05-2018 4     325.0      ORDER20      26-03-2018 | General |  | |

**Food Aggregator - Requirement 6**

Requirement 6:  
In this requirement develop a feature to find the most valuable customer in a given month.  
  
Create a **User** class with the following private attributes

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| name | String |
| email | String |
| phoneNumber | String |
| location | String |
| purchaseList | List<Purchase> |

Create / Generate appropriate Getters & Setters

Add a default constructor and a parameterized constructor to take in all attributes in the given order:

**User ( String name, String email, String phoneNumber , String location ,List<Purchase> purchaseList )**  
  
The following methods are present in the User class

|  |  |
| --- | --- |
| **Method Name** | **description** |
| static List<User> prefill() | This method returns a List of User objects. |
| static User getValuableUser(List<User> userList,String month) | This method accepts List of User objects and month name as arguments. This method returns a user object that has the maximum cumulative purchase amount in the specified month. |

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

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| id | Integer |
| price | Double |
| couponCode | String |
| purchaseDate | Date |
| user | User |

Create / Generate appropriate Getters & Setters

Add a default constructor and a parameterized constructor to take in all attributes in the given order:

Purchase(Integer id, Double price, String couponCode, Date purchaseDate, User user)  
  
The inputs to the purchase are given the order below,  
**id, price, couponCode, purchaseDate, username**  
  
Create a driver class called Main. In the main method, obtain purchase details and display the valuable user of the specified month by calling appropriate methods.  
  
Note: The output should be in the format "The valuable user of the month [month name] is [valuable user name] (Refer Input/Output specification)  
             Use "dd-MM-yyyy" for the purchaseDate attribute.

**Sample Input/Output 1:**

Enter the number of purchases:

**4**

**1,750,FIRST,05-06-2018,Rob**

**2,1500,DEAL25,05-06-2018,Brandon**

**3,2500,FIRST,07-07-2018,Joe**

**4,800,DEAL25,08-06-2018,Rob**

Enter the month:

**June**

The valuable user of the month June is Rob 

**Sample Input/Output 2:**

Enter the number of purchases:

**5**

**120,1500,CHICKEN70,06-04-2018,Oliver**

**160,2500,CHICKEN70,08-04-2018,Harry**

**190,3000,FIRST,09-04-2018,Oliver**

**225,4500,BUYFIVE,08-04-2018,Oliver**

**280,500,BUYFIVE,12-04-2018,Rob**

Enter the month:

**April**

The valuable user of the month April is Oliver

**Test Cases**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [Assign Groups](https://cognizant.e-box.co.in/problem/showTestcaseGroup/8561)   | **SNo** | **Name** | **Input** | **Output** | **Purpose** | **Sample** | | --- | --- | --- | --- | --- | --- | | 1 | t5 | 6 160,2500,CHICKEN70,06-06-2018,Harry 190,3000,FIRST,08-06-2018,Joe 225,4500,BUYFIVE,08-06-2018,Oliver 520,750,FIRST,15-07-2018,Danny 623,415,ORDER20,20-07-2018,Rob 752,1000,DEAL25,01-07-2018,Danny July | Enter the number of purchases: Enter the month: The valuable user of the month July is Danny | General |  | | 2 | t3 | 5 120,1500,CHICKEN70,06-05-2018,Oliver 160,2500,CHICKEN70,06-06-2018,Harry 190,3000,FIRST,08-06-2018,Joe 225,4500,BUYFIVE,08-06-2018,Oliver 280,500,BUYFIVE,03-03-2018,Rob March | Enter the number of purchases: Enter the month: The valuable user of the month March is Rob | General |  | | 3 | t1 | 4 1,750,FIRST,05-06-2018,Rob 2,1500,DEAL25,05-06-2018,Brandon 3,2500,FIRST,07-07-2018,Joe 4,800,DEAL25,08-06-2018,Rob June | Enter the number of purchases: Enter the month: The valuable user of the month June is Rob | General |  | | 4 | t6 | 6 160,2500,CHICKEN70,06-06-2018,Harry 190,3000,FIRST,08-05-2018,Joe 225,4500,BUYFIVE,08-06-2018,Oliver 520,750,FIRST,15-07-2018,Danny 1,750,FIRST,25-06-2018,Oliver 2,1500,DEAL25,27-06-2018,Harry June | Enter the number of purchases: Enter the month: The valuable user of the month June is Oliver | General |  | | 5 | t4 | 6 120,1500,CHICKEN70,06-05-2018,Oliver 160,2500,CHICKEN70,06-06-2018,Harry 190,3000,FIRST,08-06-2018,Joe 225,4500,BUYFIVE,08-06-2018,Oliver 280,500,BUYFIVE,03-03-2018,Rob 560,5000,FIRST,29-06-2018,Brandon June | Enter the number of purchases: Enter the month: The valuable user of the month June is Brandon | General |  | | 6 | t2 | 5 120,1500,CHICKEN70,06-04-2018,Oliver 160,2500,CHICKEN70,08-04-2018,Harry 190,3000,FIRST,09-04-2018,Oliver 225,4500,BUYFIVE,08-04-2018,Oliver 280,500,BUYFIVE,12-04-2018,Rob April | Enter the number of purchases: Enter the month: The valuable user of the month April is Oliver | General |  | |