Programming II – Test 1 (Customer)

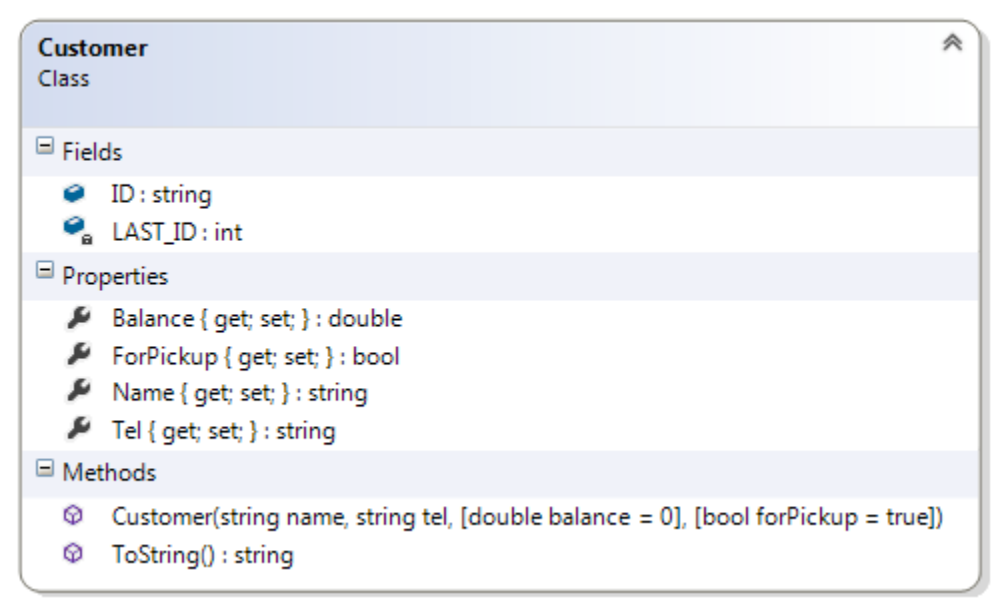
You have 90 minutes to complete all the tasks.

Your company was asked to build a contact manager for a grocery store, the software architects of your company have designed the system and your supervisor has assigned the task of coding two classes. The two classes are a Customer class and a Store class both of them are fully described below.

A test harness is provided to test your classes. You are required to match the provided output EXACTLY!

The Customer Class — 29 marks

This class is used to capture a line of information on a Store.



Fields:

3 Marks **`LAST_ID`** – this private static int represents the value to be used when creating a Customer object. It is initialized to 501. The class variable **`CURRENT_NUMBER`** is used to generate a unique string. This variable is used and updated in the constructor **`public Customer(string name, …)`**.

2 Marks **`ID`** – this string represents the id number of this Customer object. This member is set in the constructor. This field is readonly.

Properties:

All properties have public getters and private setters

2 Marks

`ForPickup` – this bool indicates if the customer requires delivery or not. This is an auto-implemented property, the getter is public and the setter is private.

2 Marks

`Name` – this string represents name of this object. This is an auto-implemented property, the getter is public and the setter is private.

2 Marks

`Balance` – this double represents the amount that is owed by this Customer. This is an auto-implemented property, the getter is public and the setter is private.

2 Marks

`Tel` – this string represents the phone number of Customers in this object. This is an auto-implemented property, the getter is public and the setter is private.

Constructor:

8 Marks

`public Customer(string name, string tel, double balance = 0, bool forPickup = true)` – This is constructor takes four argument two optional and two mandatory and does the following:

- Assigns the arguments to the appropriate properties.
- It also assigns the **`CURRENT_NUMBER`** field to the **`ID`** field (you will have to do some kind of conversion)
- And also increments it.
- The third and fourth parameters have default values.

Methods

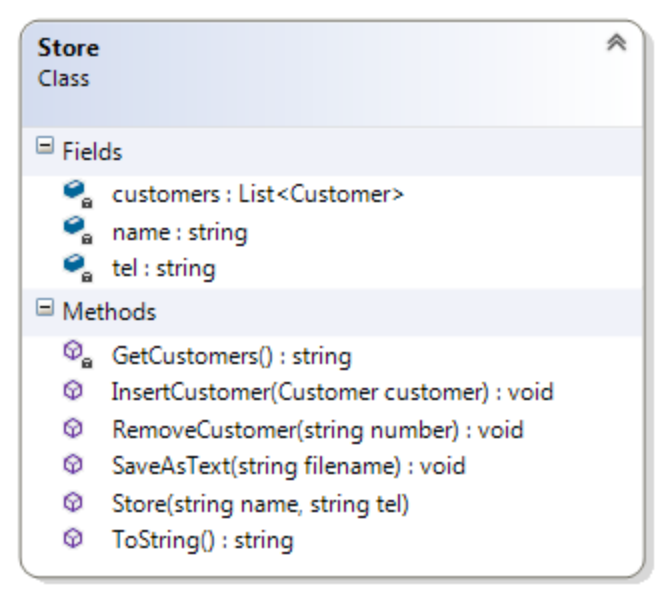
8 Marks

`public override string ToString()` – This method overrides the corresponding method of the object class to return a suitably formatted string. See the sample output for ideas on how to format your output.

This method does not display anything.

The Store Class — 38 Marks

We are going to model a Store type. There are 9 members in this class as shown in the class diagram below.



Description of class members

Fields:

All the fields are private.

3 Marks **`customers`** – this is a list of Customers. It represents a collection of Customers that comprise this Store. This is initialized at declaration. This field is private.

1 Marks **`name`** – this string represents the name of the customer. This field is private.

1 Marks **`tel`** – this string represents the phone number of the store. This field is private.

Properties:

There are no properties.

Constructor:

3 Marks `public Store(string customer, string mobile)` – This is constructor assigns the arguments to the appropriate properties.

Methods

3 Marks `public void InsertCustomer(Customer Customer)` – This public method add the argument to the field **`Customers`**.

This method does not display anything.

4 Marks

`public override string ToString()` – This is a public method overrides the corresponding method in the object class to return a stringified form of the object. In addition to the Name and Tel properties, this method uses the `GetCustomers()` method to generate a string for all the Customers. Examine the output to decide on your formatting code.

This method does not display anything.

10 Marks

`public void RemoveCustomer(string number)` – This public method removes an Customer from the collection of Customers. This method uses an appropriate loop to check each Customer in the collection. If the Number property of that Customer matches the argument then that particular Customer is removed from the collection. If the customer number is not found than throw an `Exception` object with a suitable message. [Use the method **`RemoveAt(i)`** of the list class to delete the item from the collection].

> You should not use a **foreach** loop in this method, because it iterates in a readonly fashion so you will not be able to remove it.
>
> Use either a **for** or a **while** or a **do-while** loop

This method does not display anything.

8 Marks

`private string GetCustomers()` – This is a private method that returns a string representing all the elements of the Customers collection. There is a single line for each element. This method is used in the **`ToString()`** method below to print a Store. [To get a new line use the `"\n"` sequence].

> You may use a **`foreach`** loop in this method.

This method does not display anything.

5 Marks

`public void SaveAsText(string filename)` – This is a public method saves all of the customer in this store to a text file. It relies on the **`ToString()`** method above to obtain the text. Examine the output to decide on your formatting code.

> Remember to add the necessary using statement.

This method does not display anything.

To view the resulting file correctly, drag into Visual Studio™

Test Harness

Insert the following code statements in your Program.cs file:

```
//test the Customer class
Console.WriteLine("\n*****Testing the Customer Class");
Console.WriteLine(new Customer("Yuri Gagarin", "416-123-4567"));
Console.WriteLine(new Customer("Alan Shepard", "416-345-6789", 34.56));
Console.WriteLine(new Customer("Virgil Grissom", "416-567-8901", 67.09, false));

//test the Store class
Console.WriteLine("\n*****Testing the Store Class");
Console.WriteLine(new Store("Food Basics", "647-124-5678"));

//testing InsertCustomer method of the invoice class
Console.WriteLine("\n*****Testing the InsertCustomer() for first store");
Store store0 = new Store("Fresco", "647-123-4567");
store0.InsertCustomer(new Customer("Gherman Titov", "647-123-1234", 38, false));
store0.InsertCustomer(new Customer("John Glenn", "647-123-3456", 46, false));
store0.InsertCustomer(new Customer("Scott Carpenter", "647-123-7890", 6.01));
store0.InsertCustomer(new Customer("Andriyan Nikolayev", "647-123-5498", .92,
false));
Console.WriteLine(store0);

Console.WriteLine("\n*****Testing the InsertCustomer() for second store");
Store store1 = new Store("Metro", "416-289-5000");
store1.InsertCustomer(new Customer("Pavel Popovich", "647-123-3215", 49));
store1.InsertCustomer(new Customer("Walter Schirra", "647-123-7654", 21, true));
store1.InsertCustomer(new Customer("Gordon Cooper", "647-123-9870", 78, false));
store1.InsertCustomer(new Customer("Valery Bykovsky", "647-123-3219"));
store1.InsertCustomer(new Customer("Valentina Tereshkova", "647-123-8790", 45));
store1.InsertCustomer(new Customer("Joseph Walker", "647-123-0843", 76, false));
Console.WriteLine(store1);

//testing the RemoveCustomer method of the Store class
//check the previous display to verify that atleast
//two of the item numbers are used below
Console.WriteLine("\n*****Testing the RemoveCustomer()");
store1.RemoveCustomer("508");
store1.RemoveCustomer("509");

Console.WriteLine("\n*****Saving to \"customer.txt\"");
store1.SaveAsText("customer.txt");
try
{
    store1.RemoveCustomer("508");
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}
Console.WriteLine(store1);
```

Sample Output

The following the output of a completed solution. Examine the output carefully to decide on the return value of the **`ToString()`** of the Item class and the **`ToString()`** method of the Invoice class.

```
*****Testing the Customer Class
501 Yuri Gagarin 416-123-4567 $0.00 (Pickup)
502 Alan Shepard 416-345-6789 $34.56 (Pickup)
503 Virgil Grissom 416-567-8901 $67.09 (Delivery)

*****Testing the Store Class
Food Basics tel. 647-124-5678
List of customers:

*****Testing the InsertCustomer() for first store
Fresco tel. 647-123-4567
List of customers:
 504 Gherman Titov 647-123-1234 $38.00 (Delivery)
 505 John Glenn 647-123-3456 $46.00 (Delivery)
 506 Scott Carpenter 647-123-7890 $6.01 (Pickup)
 507 Andriyan Nikolayev 647-123-5498 $0.92 (Delivery)

*****Testing the InsertCustomer() for second store
Metro tel. 416-289-5000
List of customers:
 508 Pavel Popovich 647-123-3215 $49.00 (Pickup)
 509 Walter Schirra 647-123-7654 $21.00 (Pickup)
 510 Gordon Cooper 647-123-9870 $78.00 (Delivery)
 511 Valery Bykovsky 647-123-3219 $0.00 (Pickup)
 512 Valentina Tereshkova 647-123-8790 $45.00 (Pickup)
 513 Joseph Walker 647-123-0843 $76.00 (Delivery)

*****Testing the RemoveCustomer()

*****Saving to "customer.txt"
Exception: Customer #508 was not found
Metro tel. 416-289-5000
List of customers:
 510 Gordon Cooper 647-123-9870 $78.00 (Delivery)
 511 Valery Bykovsky 647-123-3219 $0.00 (Pickup)
 512 Valentina Tereshkova 647-123-8790 $45.00 (Pickup)
 513 Joseph Walker 647-123-0843 $76.00 (Delivery)
```