Programming the .Net Framework using C#

# Billing System

A Continues homework assignment

## Introduction

The C# language offers a rich set of features. Throughout our course, we will encounter these features step by step.

The Billing System homework assignment is an ongoing project. We will start with a very simple task, and enhance it as we will learn more tools. Each step is dependent on the completion of the previous one.

For simplicity, you may use Console Application to implement your code. Try and write the classes (except for the Program test class) **as independent of the environment** (WinForm, console) as much as possible.

## Assignment #1 – Chapter 4 – Object Based Programming in C#

Define a **Customer** class, representing a customer of a telephony company. The class is defined for billing purposes – how to bill a specific customer.

Your class should consist of the following data:

* Customer name – a string
* Customer balance – a double – representing this customer due amount.
* Customer id – a unique identifier (integer). No two customers in your program should have the same id.
* You may use a static variable to help create unique ids (keep track of last created id).
* Once an id for a specific customer is set, it must never change.
* Use internal implementation method if needed.
* Define for each customer two constructors:
  + A constructor that receives a string, name as input, and initialize a customer with unique id and sets its name to the given name, and balance to 0.
  + A Constructor that receives both name and balance, and initialize a customer with unique id and sets its name to the given name, and balance to the given balance.
  + Avoid code duplication
* Define get/set properties for the customer name, balance and id. The Id property should be read only.
* Write code to print the customer details.
* Define accessibility level as needed: public, private, internal.

In the program class main, test the Customer class by creating a few instances and printing their details.

## Assignment #2 – Chapter 7 – Arrays and Strings

Add a BillingSystem class to your project. The BillingSystem should keep store of all the customers and manipulate them for billing (generate reports, balances etc).

Define a BillingSystem class with the following capabilities:

* Store customers data in an internal array
* Keep track of how many customers were added to the array
* The array size default is 100, unless otherwise defined in Billing construction.
* addCustomer(customer) method should add a customer to the billing system.
* override the BillingSystem toString method to return a string with all customers data, separated by new lines. Try to avoid creating too many string in this method. Would it be convenient to Iterate on the customers in the array using foreach statement ?
* In your main program, create a BillingSystem, create a few customers, add them to the system and print the BillingSystem.

## Assignment #3 – Chapter 8 – Object Oriented Programming

* Add the addToBalance (amount) method to the **customer** class. The method should update the current balance, depending on customer type:
  + Regular Customers – add the whole amount to customer balance
  + VIP Customer – add only 80% of the amount to customer balance
* Create two classes: RegularCustomer and VIPCustomer, as Derived classes of the Customer class. Note: not all non-VIPCustomers are RegularCustomers, we might add classes later.
* The addToBalance method should be defined in the base class. Its implementation in the base is unknown.
* Show that there is no such thing as "Customer". There are really only specific types of Customers: VIP , Regualr, etc.
* Override the toString method of the VIP customer to indicate that it’s a VIP.
* Add updateBalance(amount) method to the **BillingSystem**, iterating on all customers and activating their addToBalance(amount).
* Should you make any changes to the Billing System to acknowledge the new Customer classes? Why not?
* In your main program, create some VIPCustomers and some Regular Customers, add them to the BillingSystem, call the BillingSystem updateBalance method and see the results.

## Assignment #4 – Chapter 10 – indexers

Add indexers to the BillingSystem, allowing the retrieval of specific customers.

* One index should return the first customer with name [name] (there could be many with the same name)
* In order to be more specific, another indexer will return the customer with [id,name]
* Third index should be by location in the system (first, second, third in the list)
* Example :
* Customer cs = bs["Michal"]; // the first customer with name Michal
* Customer cs2 = bs[11,"Michal"]; // the customer with id 11 and name Michal
* Customer cs3 = bs[7]; // the seventh customer in bs

Make your indexers are for read only purposes.

## Assignment #5 – Chapter 11– Exceptions

Add Exceptions to failed situations in the BillingSystem class:

* Throw an Exception when trying to add a customer when number of customers in the system is maximal.
* Throw an Exception in the indexers above when array index is out of range
* Throw an Exception in the indexers above when the Id of a customer is as required, but the name is not.
* Use system exceptions where applicable.
* Define your own Exception to report special Exceptions.
* In your main program, use try, catch and finally to handle the Exceptions.

## Assignment #6 – Chapter 12 – Interfaces

1. Add a Sort() method to the BillingSystem, that shorts the internal customer array based on default criteria: sorting customers by name alphabetically.
   * Use Array.Sort(arr) to sort the internal array.
   * Note that Array.Sort(arr) works on a full array, null references in the array will yield an exception. You don’t have to handle this right now.
   * What should you do in the Customer class for the Sort() method to work ?
2. Advanced: In order to allow different sort Methods, define additional Sort(criteria) method in the BillingSystem.
   * Use the IComparer Interface
   * Add two classes that implement comparing customers by name, comparing customers by balance.
3. Advanced: If iterator issue was covered in class: Add ability to iterate on the BillingSystem using foreach.

## Assignment #7– Chapter 14 – Events

In order to prevent frauds, accounting problems and customer complaints, define the following event in your BillingSystem:

* **Event**: customer balance reached unreasonable limit, for example: 1000000 shekels
* **Event information**: which customer reached the threshold, and what is its balance.
* Define the EventArgs class, which will keep the above information.
* Define the **Delegate type** for the event handler
* Define the **event member in the BillingSystem** class so that clients can register to it
* In the BillingSystem, make sure that all updates to customer balance are done through one method, which checks if the updated balance does not reach the threshold. If it does, the method creates the event args object and fires the event.
* In order to simulate a problem write a method in the BillingSystem, "chargingCalls" that loops through the customers list and charge customers using the inner method above. Use random generator to get the call costs.

Define three classes that are interested to know about such cases.

* Accounting clerk want to know about this so that they can tell the CFO – a customer with a bill too large could cause the company to go out of business…
* CustomerService want to know about this so that they can call and check with the customer if someone is not misusing his phone (fraud).
* Create the above two classes, write a method in each of them that prints a message when they get the event.

In your main program, create some CustomerService and some Accounting objects, and register on the BillingSystem event.

Operate the BilllingSystem and do "chargingCalls" so that balances might reach problematic limits.

## Assignment #8– Chapter 18 – Attributes

Change the debugger display of the customer class to show only its name. Use the appropriate attribute.

If possible: make the debugger display show both the name and an indication BalanceOK if Balance < some threshold.

## Assignment #9– Chapter 19-20 – Generic Collections

BillingSystem – instead of the internal array of customers, move to a List<Customer>

## Assignment #10– Chapter 19- 20 – Generic Collections

Customer class: implement IComarable<Customer> instead of IComparer.

Try and change the IComparer classes you careated for comparison by name and balance to user IComparer<Customer>