Programming II – Test 1 (Invoice)

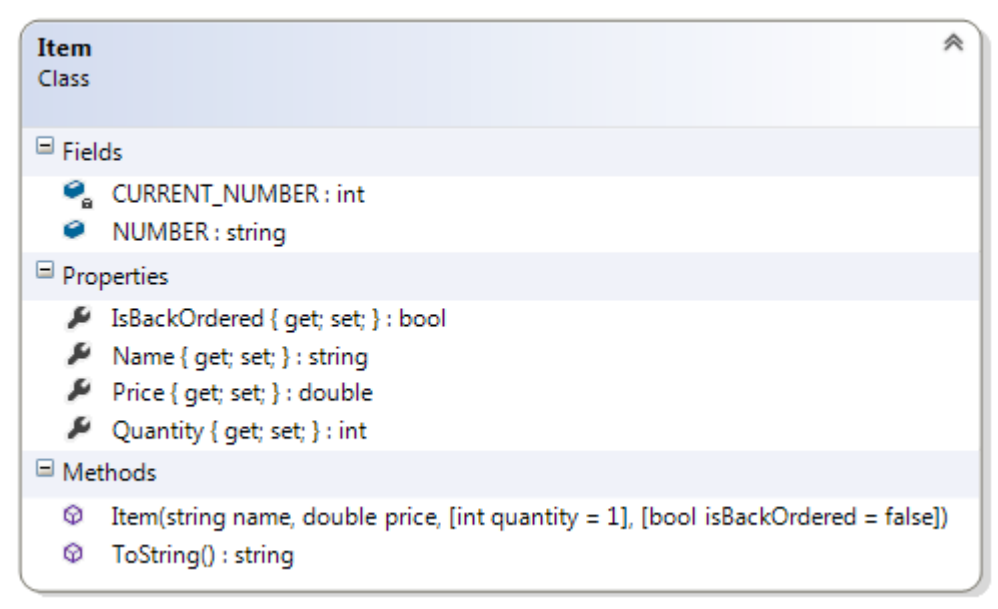
You have 90 minutes to complete all the tasks.

Your company was asked to build an accounting package, 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 an **`Item`** class and an **`Invoice`** 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 Item Class — 28 marks

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



Fields:

3 Marks **`CURRENT_NUMBER`** – this private static int represents the value to be used when creating an item object. It is initialized to 100. This variable is used and updated in the constructor **`public Item(string name, …)`**.

2 Marks **`NUMBER`** – this string represents the number of this item object. This member is set in the constructor. The class variable **`CURRENT_NUMBER`** is used to generate a unique string. This field is public and **readonly**.

Properties:

All properties have public getters and private setters

2 Marks

`IsBackOrdered` – this bool indicates if there is insufficient quantity for this line. 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

`Price` – this double represents the price of this item. This is an auto-implemented property, the getter is public and the setter is private.

2 Marks

`Quantity` – this int represents the number of items in this object. This is an auto-implemented property, the getter is public and the setter is private.

Constructor:

8 Marks

```
public Item(string name, double price, int quantity = 1, bool isBackOrdered = false)
```

– This is constructor does the following:

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

Methods

7 Marks

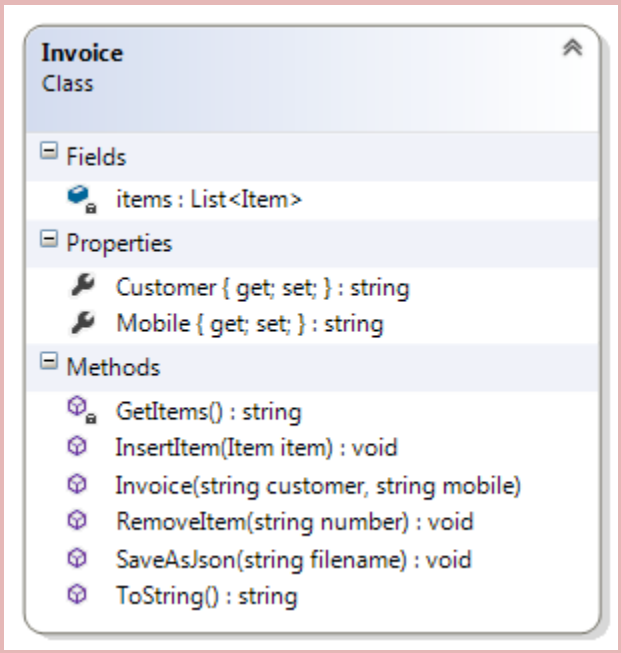
```
public override string ToString()
```

– 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 Invoice Class — 40 Marks

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



Description of class members

Fields:

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

Properties:

All the properties have public getters and private setters.

2 Marks **`Customer`** – this string represents the name of the customer. This is an auto-implemented property, the getter is public and the setter is private.

2 Marks **`Mobile`** – this string represents the mobile number of the customer. This is an auto-implemented property, the getter is public and the setter is private.

Constructor:

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

Methods

3 Marks **`public void InsertItem(Item item)`** – This public method add the argument to the field **`items`**.

This method does not display anything.

10 Marks

`public void RemoveItem(string number)` – This public method removes an item from the collection of items. This method uses a loop to check each item in the collection. If the Number property of that item matches the argument then that particular item is removed from the collection. If the invoice 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 GetItems()` – This is a private method that returns a string representing all the elements of the items collection. There is a single line for each element. This method is used in the **`ToString()`** method below to print an invoice. [To get a new line use the `"\n"` sequence].

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 Customer and Mobile properties, this method uses the **`GetItems()`** method to generate a string for all the items. Examine the output to decide on your formatting code.

This method does not display anything.

6 Marks

`public void SaveAsJson(string filename)` – This is a public method saves this invoice including all of the items to a file in json format.

> Remember to add the necessary libraries and using statement.

This method does not display anything.

Test Harness

Insert the following code statements in the **`Main()`** method of your Program.cs file:

```
//test the item class
Console.WriteLine("\n*****Testing the Item Class");
Console.WriteLine(new Item("Mouse", 5.21));
Console.WriteLine(new Item("Keybaord", 14.99, 4));
Console.WriteLine(new Item("Monitor", 124.98, 2, true));

//test the invoice class
Console.WriteLine("\n*****Testing the Invoice Class");
Console.WriteLine(new Invoice("Jean Chretien", "647-124-5678"));

//testing AddItem method of the invoice class
Console.WriteLine("\n*****Testing the AddItem() for first invoice");
Invoice inv0 = new Invoice("Kim Campbell", "647-123-4567");
inv0.InsertItem(new Item("Crucial SSD", 199.97, 5));
inv0.InsertItem(new Item("Samsung Led Monitor", 7, 4, true));
inv0.InsertItem(new Item("GeForce GTX", 28, 6, true));
inv0.InsertItem(new Item("Sapphire Radeon", 14, 12, true));
Console.WriteLine(inv0);

Console.WriteLine("\n*****Testing the AddItem() for second invoice");
Invoice inv1 = new Invoice("Brian Mulroney", "416-289-5000");
inv1.InsertItem(new Item("Netgear Router", 7, 1, true));
inv1.InsertItem(new Item("Asus Router", 21));
inv1.InsertItem(new Item("Samsung Galaxy", 56, 2, true));
inv1.InsertItem(new Item("Asus MeMO Pad", 42, 3, true));
inv1.InsertItem(new Item("Lenovo Tablet", 599.99, 1, false));
inv1.InsertItem(new Item("Ematic Tablet", 21, 4));
inv1.InsertItem(new Item("HP Elitebook", 1896.89));
inv1.InsertItem(new Item("Macbook Pro", 2300));
Console.WriteLine(inv1);

Console.WriteLine("Saving to \"invoice.json\"");
inv1.SaveAsJson("invoice.json");

//testing the RemoveItem method of the invient class
//check the previous display to verify that atleast
//two of the item numbers are used below
Console.WriteLine("\n*****Testing the RemoveItem()");
inv1.RemoveItem("108");
inv1.RemoveItem("109");

try
{
    inv1.RemoveItem("108");
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}
Console.WriteLine(inv1);
```

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 Item Class
100 - 1pcs Mouse @ $5.21
101 - 4pcs Keybaord @ $14.99
102 - 2pcs Monitor @ $124.98 (back ord)

*****Testing the Invoice Class
Jean Chretien tel. 647-124-5678
List of Items:

*****Testing the AddItem() for first invoice
Kim Campbell tel. 647-123-4567
List of Items:
 103 - 5pcs Crucial SSD @ $199.97
 104 - 4pcs Samsung Led Monitor @ $7.00 (back ord)
 105 - 6pcs GeForce GTX @ $28.00 (back ord)
 106 - 12pcs Sapphire Radeon @ $14.00 (back ord)

*****Testing the AddItem() for second invoice
Brian Mulroney tel. 416-289-5000
List of Items:
 107 - 1pcs Netgear Router @ $7.00 (back ord)
 108 - 1pcs Asus Router @ $21.00
 109 - 2pcs Samsung Galaxy @ $56.00 (back ord)
 110 - 3pcs Asus MeMO Pad @ $42.00 (back ord)
 111 - 1pcs Lenovo Tablet @ $599.99
 112 - 4pcs Ematic Tablet @ $21.00
 113 - 1pcs HP Elitebook @ $1,896.89
 114 - 1pcs Macbook Pro @ $2,300.00
Saving to "invoice.json"

*****Testing the RemoveItem()
Exception: Invoice #108 was not found
Brian Mulroney tel. 416-289-5000
List of Items:
 107 - 1pcs Netgear Router @ $7.00 (back ord)
 110 - 3pcs Asus MeMO Pad @ $42.00 (back ord)
 111 - 1pcs Lenovo Tablet @ $599.99
 112 - 4pcs Ematic Tablet @ $21.00
 113 - 1pcs HP Elitebook @ $1,896.89
 114 - 1pcs Macbook Pro @ $2,300.00
... Press enter to exit
```