Programming II – Test 1 (Patient)

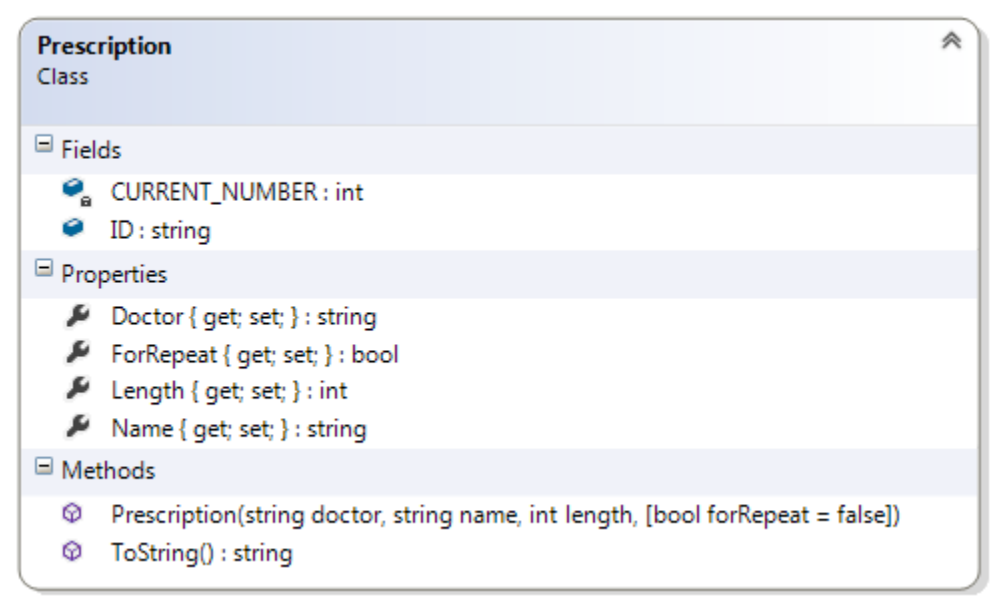
1. Test1-Customer 2. Test1-Invoice 3. Test1-Patient 4. Test1-Prescription **You have 90 minutes to complete all the tasks.**

Your company was asked to build a healthcare 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 a **`Prescription`** class and a **`Patient`** 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 Prescription Class — 30 marks

This class is used to capture the information on a prescription.



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 class variable is used (and updated) in the constructor **`public Prescription(string doctor, …)`** to generate a unique string (**`ID`**).

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

Properties:

All properties have public getters and private setters

2 Marks **`ForRepeat`** – this bool indicates if this prescription may be repeated. This is an auto-implemented property, the getter is public and the setter is private.

2 Marks **`Doctor`** – this string represents name of the doctor who prescribed this medication. This is an auto-implemented property, the getter is public and the setter is private.

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

2 Marks **`Length`** – this int represents the number of days that this prescription is supposed to last. This is an auto-implemented property, the getter is public and the setter is private.

Constructor:

10 Marks **`public Prescription(string doctor, string name, int length, bool forRepeat = false)`** – This is constructor takes four arguments (the last one having a default value of false). It does the following:

- Assigns all 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)
- It increments **`CURRENT_NUMBER`**.
- The fourth argument has a default value.

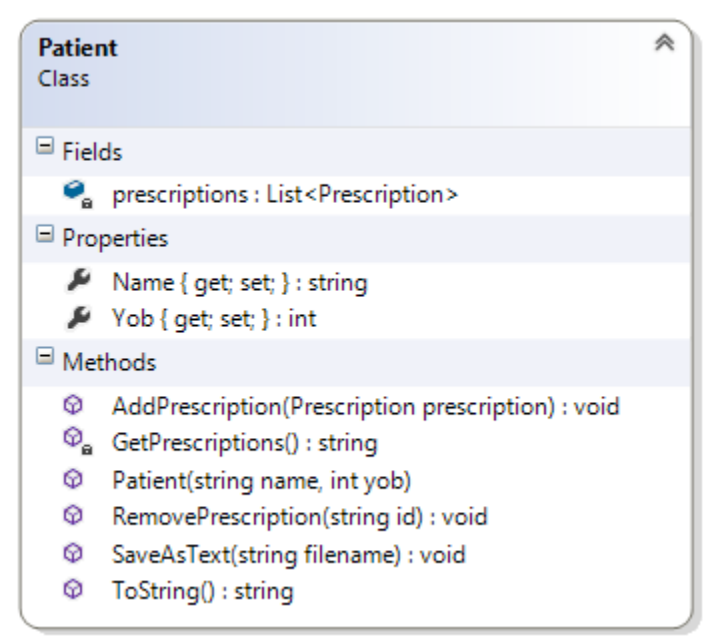
Methods

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 Patient Class — 38 Marks

We are going to model an Invoice type. There are 8 members in this class as shown in the class diagram below:



Description of class members

Fields:

4 Marks **`prescriptions`** – this is a list of prescription. It represents a collection of items that comprise this patient. This is initialized at declaration. This field is private.

Properties:

All the getters are public and the setters are private.

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

2 Marks **`Yob`** – this int represents the year of birth of this patient. This is an auto-implemented property, the getter is public and the setter is private.

Constructor:

3 Marks **`public Patient(string name, int yob)`** – This is constructor assigns the arguments to the appropriate properties.

Methods

3 Marks **`public void AddPrescription(Prescription prescription)`** – This public method add the argument to the field **`prescriptions`**.

This method does not display anything.

5 Marks

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

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

This method does not display anything.

Do not use the **`string.Join()`** method. You must use a loop to build the required output.

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

> Remember to add the necessary using statement.

This method does not display anything.

4 Marks

`public void SaveAsText(string filename)` – This is a public method saves all of the prescriptions for this patient to a file in plain text format.

This method does not display anything.

Test Harness

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

```
//test the prescription class
Console.WriteLine("\n*****Testing the Prescription Class");
Console.WriteLine(new Prescription("Nika", "Asprin", 21));
Console.WriteLine(new Prescription("Khan", "Amoxicillin", 14, true));

//test the patient class
Console.WriteLine("\n*****Testing the Patient Class");
Console.WriteLine(new Patient("Justin Trudeau", 1989));

//testing AddPrescription method of the patient class
Patient pat0 = new Patient("Stephen Harper", 1990);
Console.WriteLine("\n*****Testing the AddPrescription() method");
pat0.AddPrescription(new Prescription("Pershad", "Coumadin", 21));
pat0.AddPrescription(new Prescription("Pershad", "Metformin", 7, true));
pat0.AddPrescription(new Prescription("Tapia", "Ventolin", 28, true));
pat0.AddPrescription(new Prescription("Nika", "Cipralex", 14, true));
Console.WriteLine(pat0);

Patient pat1 = new Patient("Paul Martin", 1993);
Console.WriteLine("\n*****Testing the AddPrescription() method");
pat1.AddPrescription(new Prescription("Khan", "Tylenol", 7, true));
pat1.AddPrescription(new Prescription("Zouri", "Synthroid", 21));
pat1.AddPrescription(new Prescription("Filotti", "Ramipril", 56, true));
pat1.AddPrescription(new Prescription("Pershad", "Pravastatin", 42, true));
pat1.AddPrescription(new Prescription("Filotti", "Metformin", 7, true));
pat1.AddPrescription(new Prescription("Lac", "Metformin ", 190));
pat1.AddPrescription(new Prescription("Li", "Fluvastatin", 21));
pat1.AddPrescription(new Prescription("Nika", "Coumadin", 21));
Console.WriteLine(pat1);

Console.WriteLine("\n*****Saving to \"patient.txt\"");
pat1.SaveAsText("patient.txt");

//testing the RemovePrescription method of the patient class
Console.WriteLine("\n*****Testing the RemovePrescription() method");
pat1.RemovePrescription("108");
pat1.RemovePrescription("109");
try
{
    pat1.RemovePrescription("108");
}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}
Console.WriteLine(pat1);

Console.Write("... Press enter to exit");
Console.ReadLine();
```

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 Prescription class and the **`ToString()`** method of the Patient class.

```
*****Testing the Prescription Class
#100 Asprin prescribed by Nika for 21days (No repeat)
#101 Amoxicillin prescribed by Khan for 14days (Repeat)

*****Testing the Patient Class
Justin Trudeau yob: 1989
List of prescription:

*****Testing the AddPrescription() method
Stephen Harper yob: 1990
List of prescription:
 #102 Coumadin prescribed by Pershad for 21days (No repeat)
 #103 Metformin prescribed by Pershad for 7days (Repeat)
 #104 Ventolin prescribed by Tapia for 28days (Repeat)
 #105 Cipralex prescribed by Nika for 14days (Repeat)

*****Testing the AddPrescription() method
Paul Martin yob: 1993
List of prescription:
 #106 Tylenol prescribed by Khan for 7days (Repeat)
 #107 Synthroid prescribed by Zouri for 21days (No repeat)
 #108 Ramipril prescribed by Filotti for 56days (Repeat)
 #109 Pravastatin prescribed by Pershad for 42days (Repeat)
 #110 Metformin prescribed by Filotti for 7days (Repeat)
 #111 Metformin  prescribed by Lac for 190days (No repeat)
 #112 Fluvastatin prescribed by Li for 21days (No repeat)
 #113 Coumadin prescribed by Nika for 21days (No repeat)

*****Saving to "patient.txt"

*****Testing the RemovePrescription() method
Exception: Prescription 108 does not exist
Paul Martin yob: 1993
List of prescription:
 #106 Tylenol prescribed by Khan for 7days (Repeat)
 #107 Synthroid prescribed by Zouri for 21days (No repeat)
 #110 Metformin prescribed by Filotti for 7days (Repeat)
 #111 Metformin  prescribed by Lac for 190days (No repeat)
 #112 Fluvastatin prescribed by Li for 21days (No repeat)
 #113 Coumadin prescribed by Nika for 21days (No repeat)
```