



**MALMÖ HÖGSKOLA**

**Malmö University  
School of Technology**

Programming Using .NET, Basic Course (C# and VB)

## OOP and Inheritance

### Assignment 6

### Apu's Sweets

**Mandatory**

[Farid Naisan](#)

University Lecturer  
Department of Computer Science

## Summer special using inheritance

### 1 Objectives

The main objectives of this assignment are to implement the second essential concept of object-orientation, **inheritance**, with a simple project. Inheritance addresses reuse of code. The following topics are included in this assignment:

- Inheritance, the "Is a" relation between objects
- Dynamic binding
- Working more with encapsulation and using properties

### 2 Description

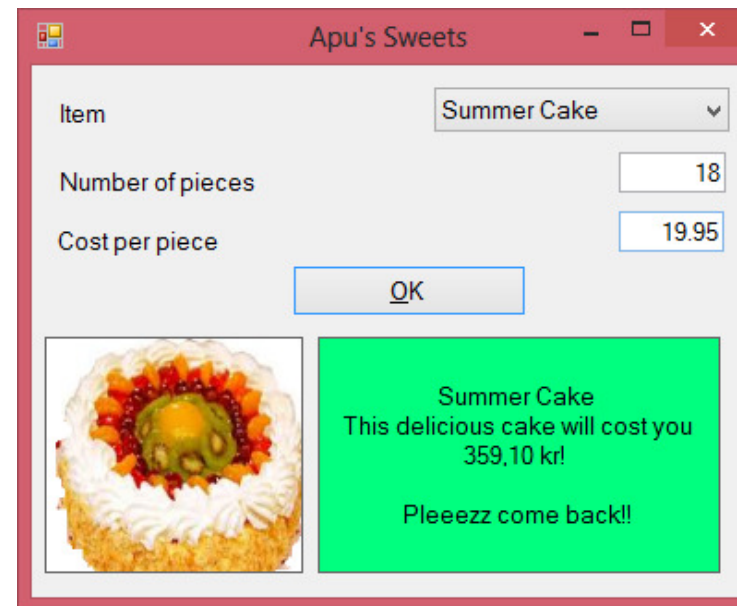
This assignment can be done as

- **Alternative A:** you do the project APU's Cakes and Cookies specified below.
- **Alternative B:** you can think of an idea for an application and program it implementing inheritance
- In both cases, you have to follow the requirements described later in this document.

### 3 Apu's Sweets

As the summer is approaching at the time of this writing, everyone is preparing for it, especially those who have business and want to make money. Apu has a business idea for this season. He has established a small temporary road-side shop, Apu's Sweets (ASS), in Lund.

ASS sells only a few products, a special mix of cookies and a few cake sorts. What he needs is a program that takes the customer's order and calculates the total price to pay as illustrated by the run example shown here. Remember that the user of the application is Apu himself or an employee working for him in the shop.

#### Details:

- 3.1 ASS's products are limited to those sorts defined by the **enum** ProductType.
- 3.2 The cakes are sold by **pieces** but cookies are sold by **weight** because the mixture contains cookies of different sizes and weights.
- 3.3 To keep things simple, only one type of product can be ordered at a time. The application writes a receipt on the screen as in the previous figure when the OK button is pressed.

```
public enum ProductType
{
    Spring_Cake_Special,
    Summer_Cake,
    Cookies,
    Princess_Cake,
    FruitCake,
}
```

## 4 Specifications and Requirements for a passing grade (G or ECT C)

### Inheritance

- 4.1 All products have a **name** and a **price** per unit.
- 4.2 Each product type is to be represented by a separate class.
- 4.3 Inheritance must be implemented. The application should have at least one base class and two subclasses.
- 4.4 The base class should have at least 2 private fields.
- 4.5 The base class should have a parameterized constructor with at least one parameter. It should not have any default constructor.

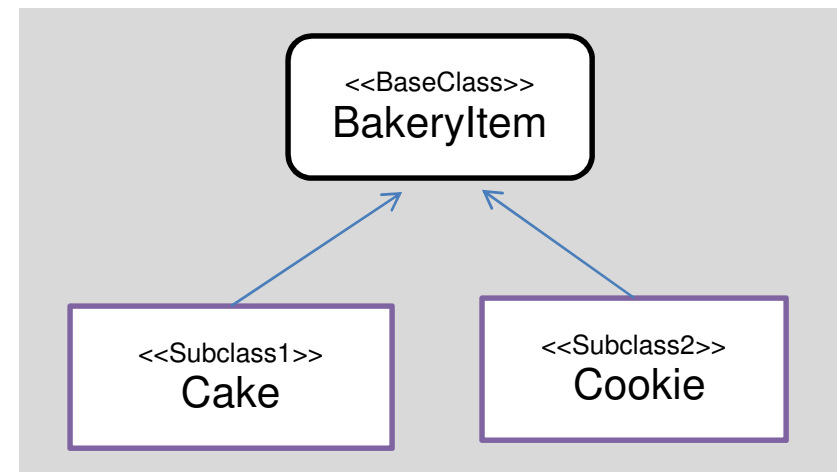
**Remember that this requirement is set only for the purpose of practicing with subclasses calling the base class' constructor.**

However, if you write a constructor in the sub classes that calls a parameterized constructor in the base class, writing a default constructor is permitted.

Hint: A base class' constructor can be called from the subclass as the example below:

```
public Cake(string name, double price):base(name, price) //constructor in the Cake class
```

- 4.6 Subclasses should have at least one field that is different from other subclasses.



- 4.7 All classes should override the **ToString** method formatted with values saved in the object's fields including those in the base class. Subclasses should make use of the base class' **ToString** method.
- 4.8 Showing images is **optional**.
- 4.9 Write **set** and **get** properties to provide access to fields of the class.
- 4.10 **Use of auto-properties is not allowed in any class.** The set properties should validate the value of "**value**"; As an example, suppose that you have a field named **m\_name**:

## C#

```
public string Name
{
    get
    {
        return m_name;
    }
    set
    {
        if (!string.IsNullOrEmpty(value))
            m_name = value;
    }
}
```

## VB

```
Public Property Name() As String
    Get
        Return m_name
    End Get
    Set(value As String)
        If Not String.IsNullOrEmpty(value) Then
            m_name = value
        End If
    End Set
End Property
```

**Note:** This requirement is mainly for the purpose of letting you exercise with properties in their basic forms. It is important to understand how properties can be used to take full control of the value that is returned or accepted.

## GUI

- 4.11 The application must have a graphical user interface. Design your GUI according to your application's needs for input and displaying output. Determine which values should come from user as input and which values may be expected from the application by the user.
- 4.12 The application should create the correct type of object (of subclasses) based on the user interaction. Design your GUI so it allows you to create an object of all the sub classes included in your application. Based on the class diagram shown earlier, the user should be able to select an object of Subclass1 or Subclass2 and your application should then create the corresponding object type.

- 4.13 Handle input, output and calculations in an organized way using a method for every task, as you have learned in the previous assignments.
- 4.14 All user interactions are to be handled of the Form object, **MainForm**, and all logics should be handled of other classes, as in our previous assignments. No other classes than the **MainForm** should have any interaction with the user.
- 4.15 **Optional:** Use a **TabControl** to group your controls in different tabs.

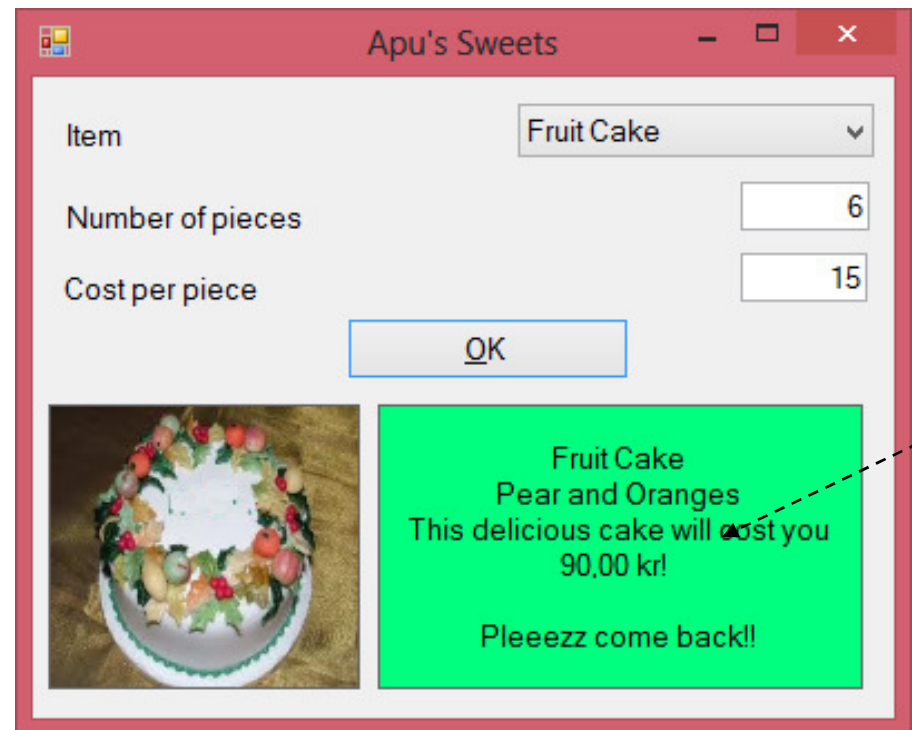
## Project

- 4.16 Organize your project files and place related files in subfolders.
- 4.17 Document your classes and methods with xml-compatible comments.
- 4.18 Following a good quality standard and make sure that your application can be compiled and it works well and does not crash for wrong input before submitting.
- 4.19 If you program your own idea, include a class diagram in your project using VS (or other software). Write a brief note about what your application does and how it works. You may use a text file included in your application or just write a comment when submitting your project.

## 5 Specifications and Requirements for a higher grade (VG or ECT A and B)

The bakery has a special cake, the Fruit Cake containing two fresh fruits as a season's surprise. The fruits are to be randomized for every order by the program among a number of fruits such as apples, pears, pineapples strawberries, raspberries, blueberries (my favorite), etc. No objects are required to be written for the fruits. They can be encapsulated as items inside an **enum** type, and presented as strings to the user.

- 5.1 All the requirements for a Pass grade, as listed above, are valid for this part as well.
- 5.2 Establish one more level in the inheritance hierarchy and create a subclass that inherits one of the subclasses as its base class, as illustrated in the figure below (Subclass3).



5.3 Use at least one virtual method in one of the base classes and override it in a subclass.

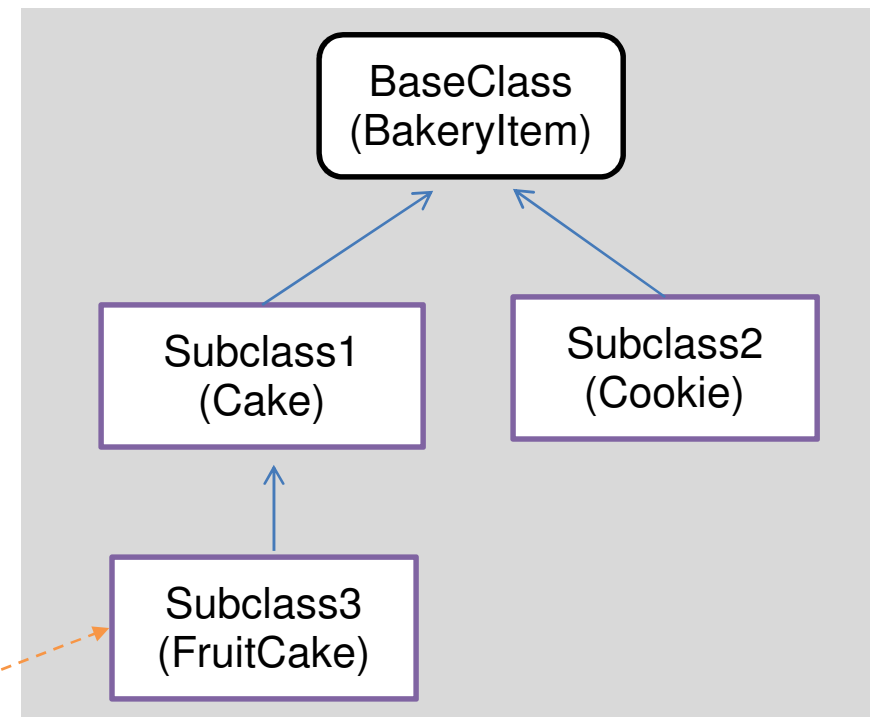
**Optional:** As a good practice, you can write the virtual method in the first **BaseClass (BakeryItem)** and override it in the **Subclass3 (FruitCake)**.

5.4 Use dynamic binding (see lecture material) in creating objects.

5.5 **Optional:** Use a **PictureBox** on the GUI to show an image related to the object type you are currently working with. You can download images from the Internet or draw them yourself. You may use an **ImageList** to hold the images or put the images in your resource file. You can even upload images at run time. There is a separate document about using resources in Visual Studio, available for downloading in the module.

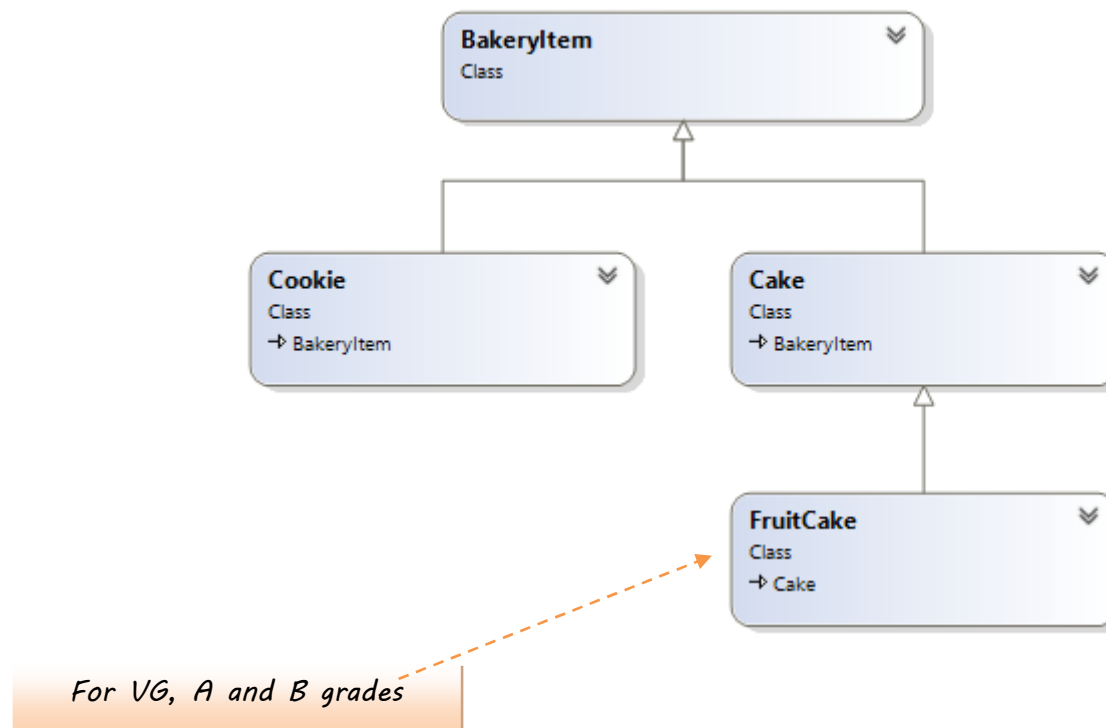
## 6 Submission

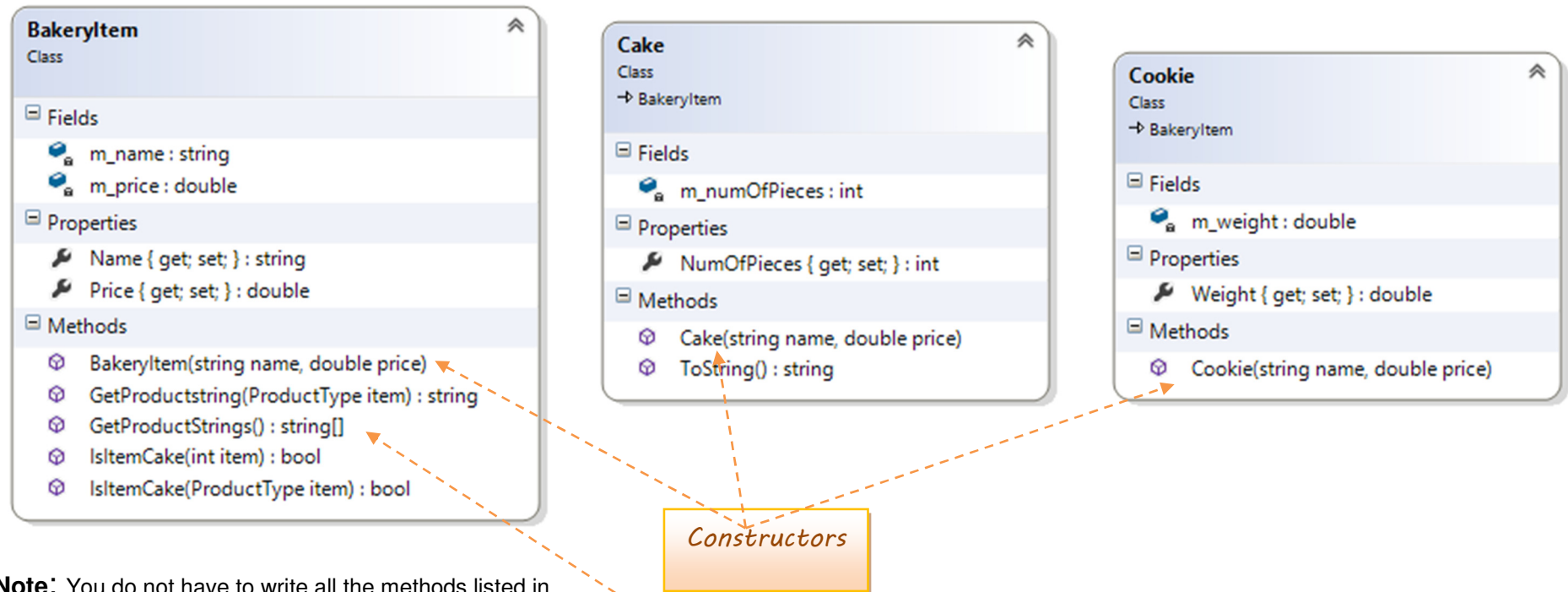
Submission is done as previous assignments.



## 7 Help and guidance

Every topic addressed in this assignment is covered in the lectures and exercises in the module. Refer to these materials whenever you feel unsure about a subject. Make also use of the forums. The class diagram below might give you some clue, but you don't have to follow this solution in every detail or write every single method listed in the class specifications.





**Note:** You do not have to write all the methods listed in the figures. You can write your own methods.

*Prepare and return an array of strings, where every string contains a textual representation of the members of the enum ProductType but without the "\_" char.  
(This method calls the method GetProductString which does the same thing but for one specific enum member).*

Good Luck!

**Farid Naisan,**  
Course Responsible and Instructor