C# for Beginners

Windows Desktop Applications using Windows Forms

.NET

Contents:

- Windows Forms and Controls
- Event-driven programming
- Properties, Events and Event Handlers

Farid Naisan, University Lecturer, Malmö University, farid.naisan@mau.se

Windows Applications

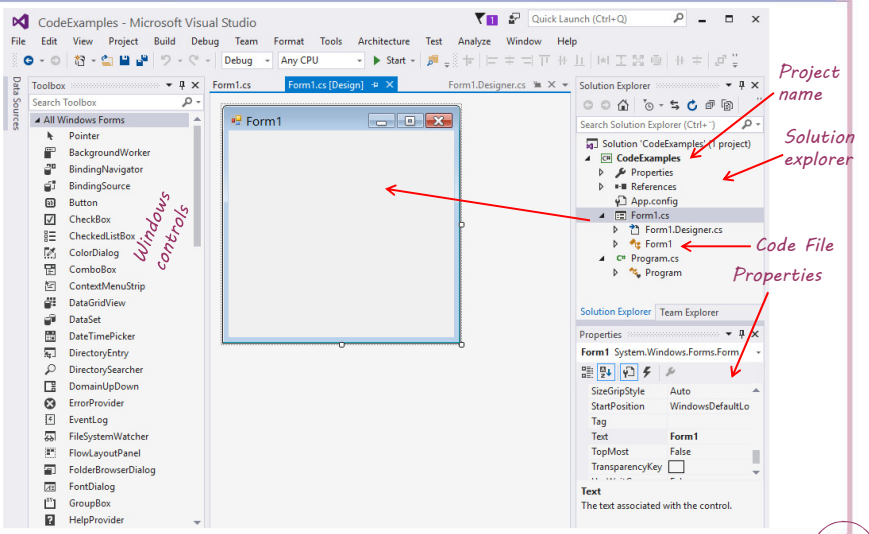
.NET

- .NET includes all the classes and features needed to develop desktop applications.
- It contains many namespace to categorize related classes and types.
- System.Windows.Forms is a namespace hosting a large selection of types for designing and writing applications with graphical user interface (GUI).
- The GUI components are grouped as **Forms** and **Controls**. They are all windows based and have a parent class called System.Windows.Forms.Control.
- A Control is a graphical component with many features.
  - TextBox, Label, Button, Icon are all controls.

Windows Forms

- A Form is a class and we need to create an object of this class before we can access its feature-rich members.
- You can think of a form as a container for other controls, although you write text or draw graphics directly on its surface.
  - However, there are other components that are particularly designed for such purposes.
- When you create a Windows Forms Application in VS, an object of the Form class is created by VS with the default name “Form1”, ready for you to begin designing your GUI.

An Example Project

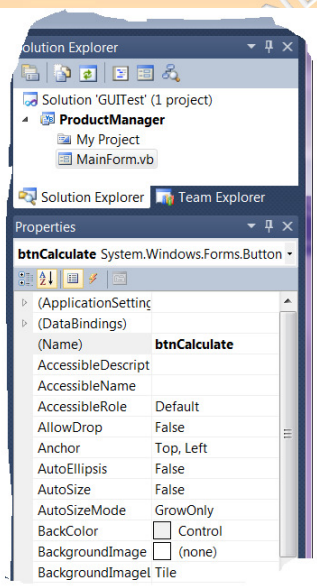


Windows Forms – starting object

- When working with GUI applications using Visual Studio, Visual Studio creates automatically a start up class with a Main method (named Program.cs by default).
- In this method, it creates Form1 which becomes the staring object meaning that when you run the application, Form1 will be loaded and displayed to the user.
- Every application usually has multiple Forms. One of these has to be selected as the start up Window. If you wish to select another Form as your starting object open the folder Properties inside VS, in the Project window.
- Designing a Windows Forms Application is very easy using Visual Studio, but it can be developed using an ordinary text editor like Notepad and of course the .NET SDK.

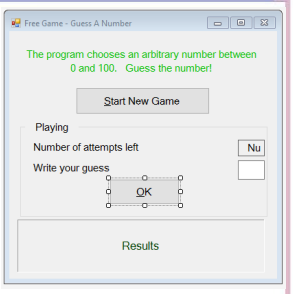
Working with Forms and Controls

- All Forms and Controls are instances of classes
- They have:
  - Fields
  - Properties and methods
  - Events
  - Event-handlers
- An event is an action that occurs during the program usage at run time.
  - Clicking on a button fires the Click



Event-handler method

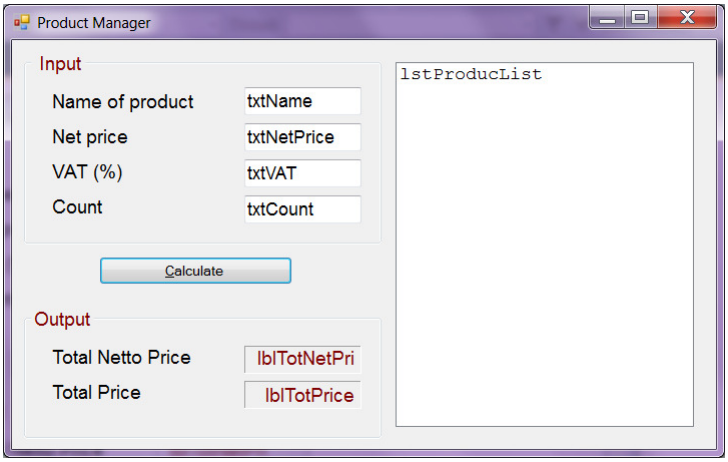
- An event-handler is a method that is connected to an event inside the code and is automatically executed when the event occurs at run time.
- The **btnOK_Click** method below is an example of such a method. It is connected to the OK button. btnOK is the Name-property of the button




```
//Every time the user clicks the OK button, this method gets executed.
//Sender: is the object that notifies about the clicking, here it is
//the btnOK button.  The parmeter e is a structure that contains
//information about the clicking (whether right or left button is used)
1 reference
private void btnOk_Click (object sender, EventArgs e)
{
    //All code here will be executed automatically.
}
```

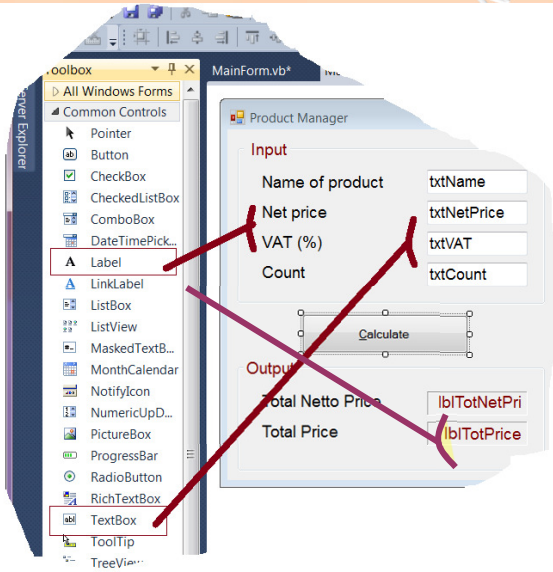

GUI Example – The Product Manager

- Same naming rules as variables apply also for controls.



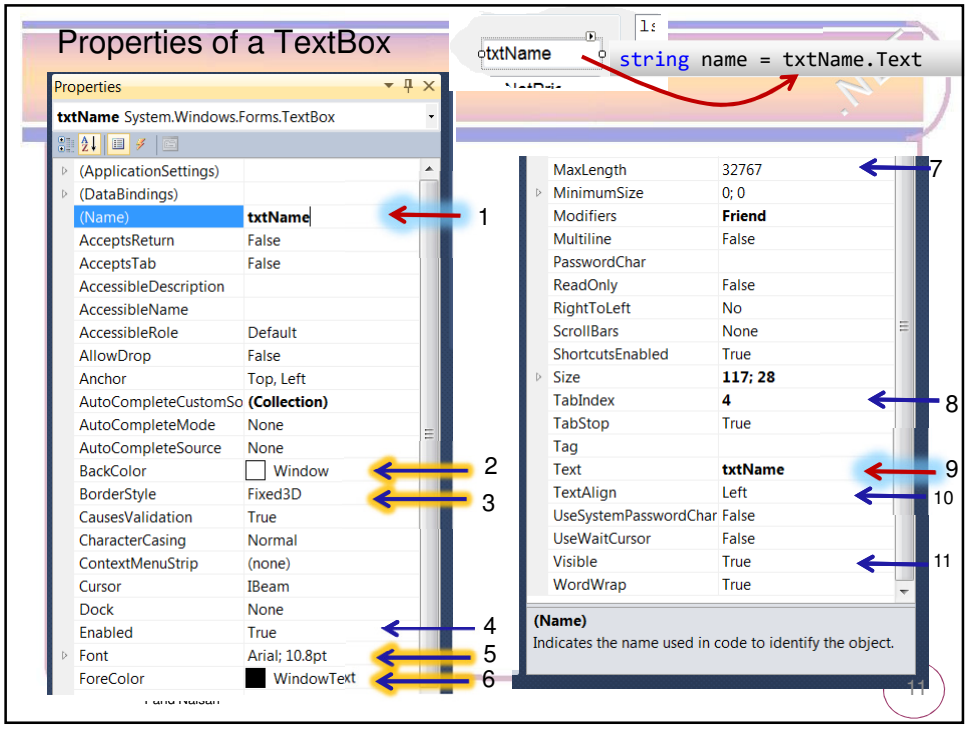
Textboxes and labels

- TextBoxes are used for editable text (input)
- Labels are used for read-only text (heading, info, output).



Properties

- Every Control has many properties that you can affect the look and feel of the control..
- Some of the properties can be changed only at design time, some only at run-time (from code) and some either way.
- Certain properties are not possible to change. These are read-only.
- Each property has a value of a certain type. The **Text** property (ex textBox1.Text) for example is a string, while the **Enabled** and **Visible** properties are Booleans.
- Some properties have values that are constants and should be chosen from an Enumeration.



Methods

- Forms and Controls have also many useful methods that you can use in your code.
- For example some controls have a method called **Focus** which means that it is ready to receive the user's input.
- The focus can be set to a control in code using the Focus method:

```
txtNetPrice.Focus();  //method call
```

- This method sets the focus to the textbox named txtNetPrice.
- Note:
  - Properties are used exactly as variables.
  - txtProductName.Text = “Egg”; //no parenthesis after .Text
  - Methods are used as ordinary method calls, as above.

Event-Driven Programming

- An event is an action that takes place within a program, such as the clicking of a button.
- An Event is an action performed by the user interacting with an application, or by code in your application.
- Applications respond to an Event by providing methods that are automatically called.
- Events and event-handlers are the key concepts in the event-driven mechanism.
- An event-handler method is connected to an event easily using VS.

Events and event-handler methods

- All Visual C# .NET controls are capable of detecting a set of predefined events.
- You can handle an event by writing code in a special method connected to a control.
- To handle an event means that you will write code that will instruct the program what actions to take whenever a specific event is triggered.

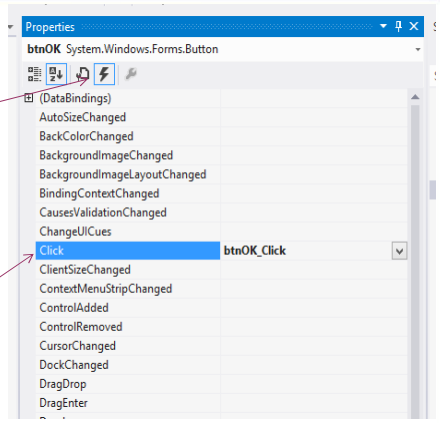
```
//Event-handler method connected to the Click event
//of the button btnOK.
//Sender: the object sending the event
//e: object containing relevant info about the event.
private void btnOK_Click(object sender, EventArgs e)
{
    //Write code to do things when the OK button is clicked.
}
```

Events and event handlers

- Forms and Controls decide which event they make available.
- Events have names. The click-event of a control has the name **Click**.
- Other examples are
  - **Load, Unload, MouseDown, MouseUp, MouseMove, MouseDoubleClick, KeyPress, KeyDown, KeyUp, SizeChanged**, etc.
- Each event has also a set of parameters that are made available to the event-handler, as in the previous slide.

Example – Button Events

- To see all events for a certain control, select the control, click on the lightning icon.
- To create an event-handler method for a control, double-click on the event.
- The figure shows the Click-event for a button.

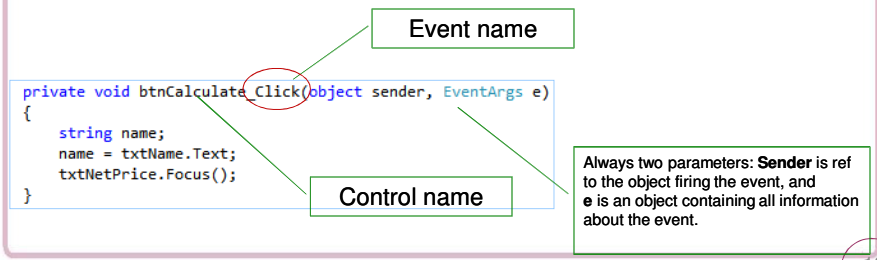


Firing Events

- A large number of events are fired by user actions.
- But there are many events that are fired only when the control is modified by the source code:
  - **TabIndexChanged, BackgroundImageChanged, CursorChanged**, etc.
- A few events are fired by system-level changes, such as the **SystemColorsChanged** event, fired when the system-wide color scheme is changed through the control panel.

Event Handlers

- An event handler is a void method that will be executed when an event is fired. You can put custom code inside this method.
- An event handler is to be written with a special signature syntax:



Event Delegates

- A delegate is a type in .NET that can store the address of one method or a number of methods in the same or different objects.
- Delegates are used very frequently by Windows Forms.
- VS takes care of all the necessary coding to let a delegate remember for example when to invoke a handler method when an event is triggered (clicking on a button).
- All event-handler methods are invoked by delegates.
- VS uses delegates to attach an event-handler to an event of a control.
- The delegate remembers the method and calls it when the attached event is fired.

Use of Delegate - example

- The code in the figure is generated by VS when you double-click on the button btnOK at design time.

```
this.btnOK.Click += new System.EventHandler(this.btnOK_Click);
```

- The method **btnOK_Click** of the **MainForm** will be automatically invoked when the **btnOK** fires the **Click** event, when the user clicks on the button.
- The **+=** symbol adds the method in the list of methods that their objects want them to be executed when the Click event is fired.
- This way objects ”subscribe” to an event of another object.

Events in The Life Cycle of a Form

Event	Occurs …
Move	when the form is being created but before the Load event.
Load	when the Form is created but before it is displayed for the first time.
VisibleChanged	the value of the Visible property changes.
Activated	when the form is activated by the source code or by the user.
Shown	when the Form is displayed for the first time.
Paint	when the control is redrawn.
Deactivated	when the form loses focus and is not the active form.
Closing	just before the form is closed.
Closed	when the form is being closed.

Summary

- A Windows Form is used to design a Graphical User Interface (GUI).
- There are numerous controls under the System.Windows.Forms namespace available for a programmer.
- These are shown in the ToolBox in VS.
- Write event-handler methods to your controls to perform tasks.
- A handler method connected to an event of a control will automatically be executed when the event is fired by the control.
  - The event is fired as a result of user interaction or if the event is triggered from somewhere in the code.
  - When you click on a button on the form, the Click event will be fired. If you have written code in the buttons Click-event handler method, that code will be executed then.