Chapter 22

Introduction to Windows Forms

Introduction to Windows Forms

Objectives

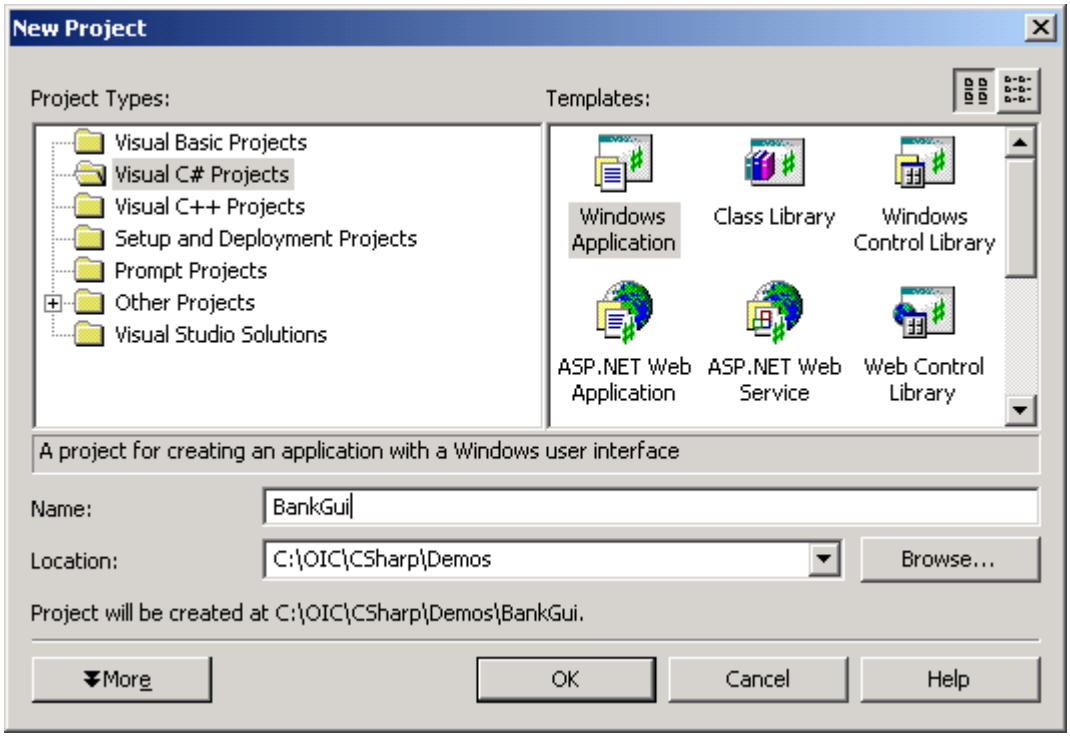
After completing this unit you will be able to:

- **Use Visual Studio .NET to implement a simple graphical user interface with Windows Forms and basic controls.**
- **Explain the principles of event handling in Windows Forms and implement handlers for simple control events.**

Creating a Windows Forms App

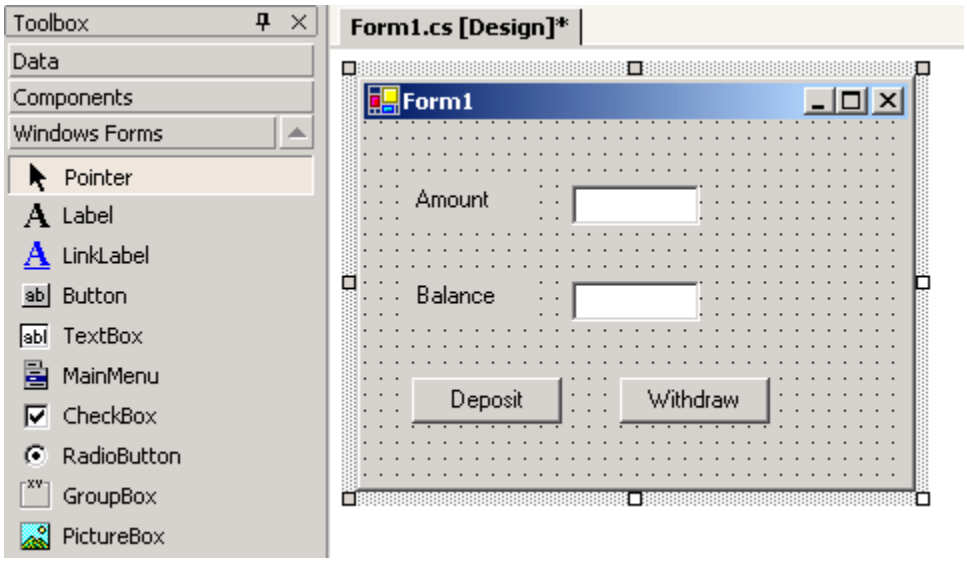
- **It is easy to create a Windows Forms application using the Forms Designer in Visual Studio .NET.**
  - This same Forms Designer is available to all .NET languages.

1. Create a new C# project **BankGui** of type Windows Application in the Demos folder.



Windows Forms Demo (Cont'd)

2. From the toolbox drag two labels, two textboxes, and two buttons to the form.



3. Enter property values for the textboxes and buttons:

Name	Text
txtAmount	(blank)
txtBalance	(blank)
cmdDeposit	Deposit
cmdWithdraw	Withdraw

4. Add event handlers for the buttons by double clicking on each button.

Windows Forms Demo (Cont'd)

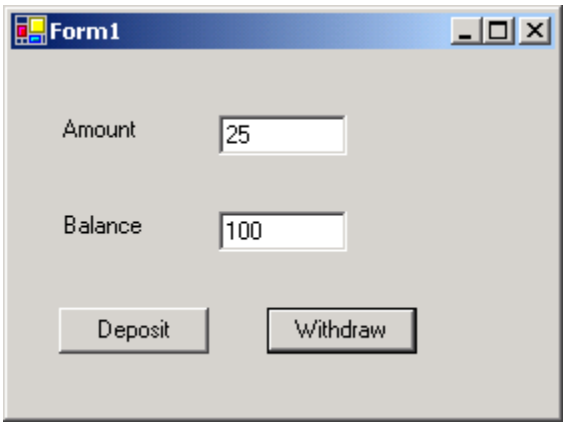
5. Add the following code:

```
public class Form1 : System.WinForms.Form
{
  ...
  public Form1()
  {
    txtAmount.Text = "25";
    txtBalance.Text = "100";
  }
  ...
  protected void cmdWithdraw_Click (
             object sender, System.EventArgs e)
  {
    int amount = Convert.ToInt32(txtAmount.Text);
    int balance = Convert.ToInt32(txtBalance.Text);
    balance -= amount;
    txtBalance.Text = Convert.ToString(balance);
  }

  protected void cmdDeposit_Click (
                 object sender, System.EventArgs e)
  {
    int amount = Convert.ToInt32(txtAmount.Text);
    int balance = Convert.ToInt32(txtBalance.Text);
    balance += amount;
    txtBalance.Text = Convert.ToString(balance);
  }
  [STAThread]
  public static void Main(string[] args)
  {
    Application.Run(new Form1());
  }
}
```

Windows Forms Demo (Cont'd)

6. Build and run the application. It should behave like a standard Windows application.



The completed demo is in the **BankGui** directory for this chapter.

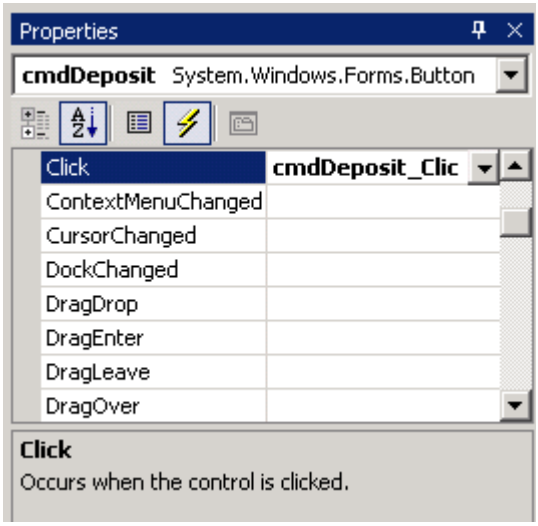
Windows Forms Event Handling

- **GUI applications are event-driven, in which the application executes code in response to user events, such as clicking the mouse, choosing a menu item, etc.**
- **Each form or control has a predefined set of events.**
  - For example, every button has a **Load** event.
- **Windows Forms uses the .NET *event* model, which uses *delegates* to bind events to the methods that handle them.**
  - Windows Forms typically uses *multicast* delegates.
  - A multicast delegate maintains a list of the methods it is bound to.
  - When an event occurs in an application, the control raises the event by calling the delegate for that event.
  - The delegate then calls all the methods it is bound to.
- **C# provides the overloaded += operator for adding a delegate to an event.**

```
this.cmdDeposit.Click +=
   new System.EventHandler(this.cmdDeposit_Click);
```

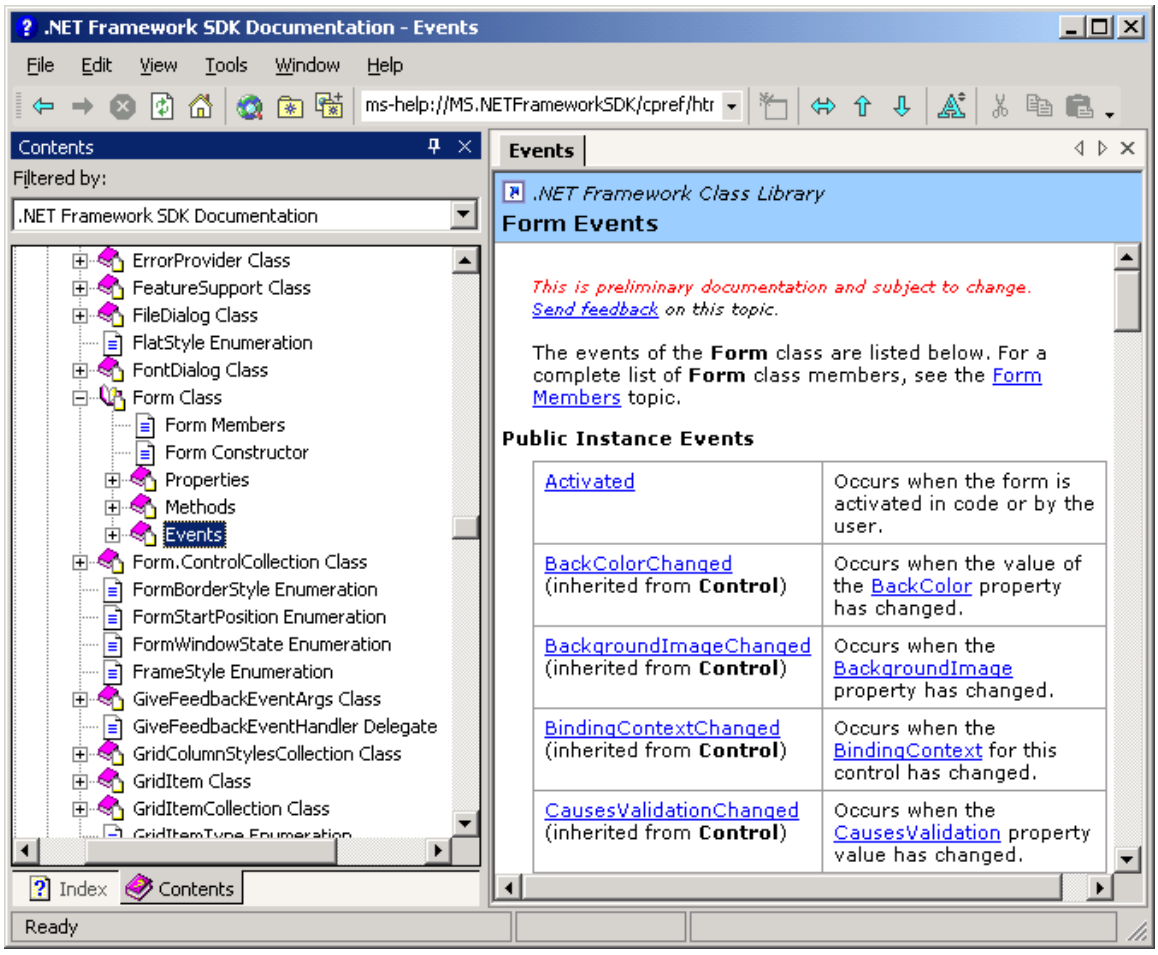
Add Events for a Control

- **Every control has a *primary* event, and you can add a handler for this event by double-clicking on the control in the Visual Studio .NET Form Designer.**
  - We added a handler for the Deposit and Withdraw buttons in this way.
- **You can add other events by selecting the  in the Properties window.**



Events Documentation

- **You can find all the events associated with a class in the .NET Framework Reference.**
  - The screen shot shows the predefined events associated with the **Form** class.



Summary

- **With Visual Studio.NET you can implement a simple graphical user interface with Windows Forms and basic controls.**
- **Windows Forms uses the .NET event architecture to handle user interfa**