Chapter 9

Windows Applications using Winforms

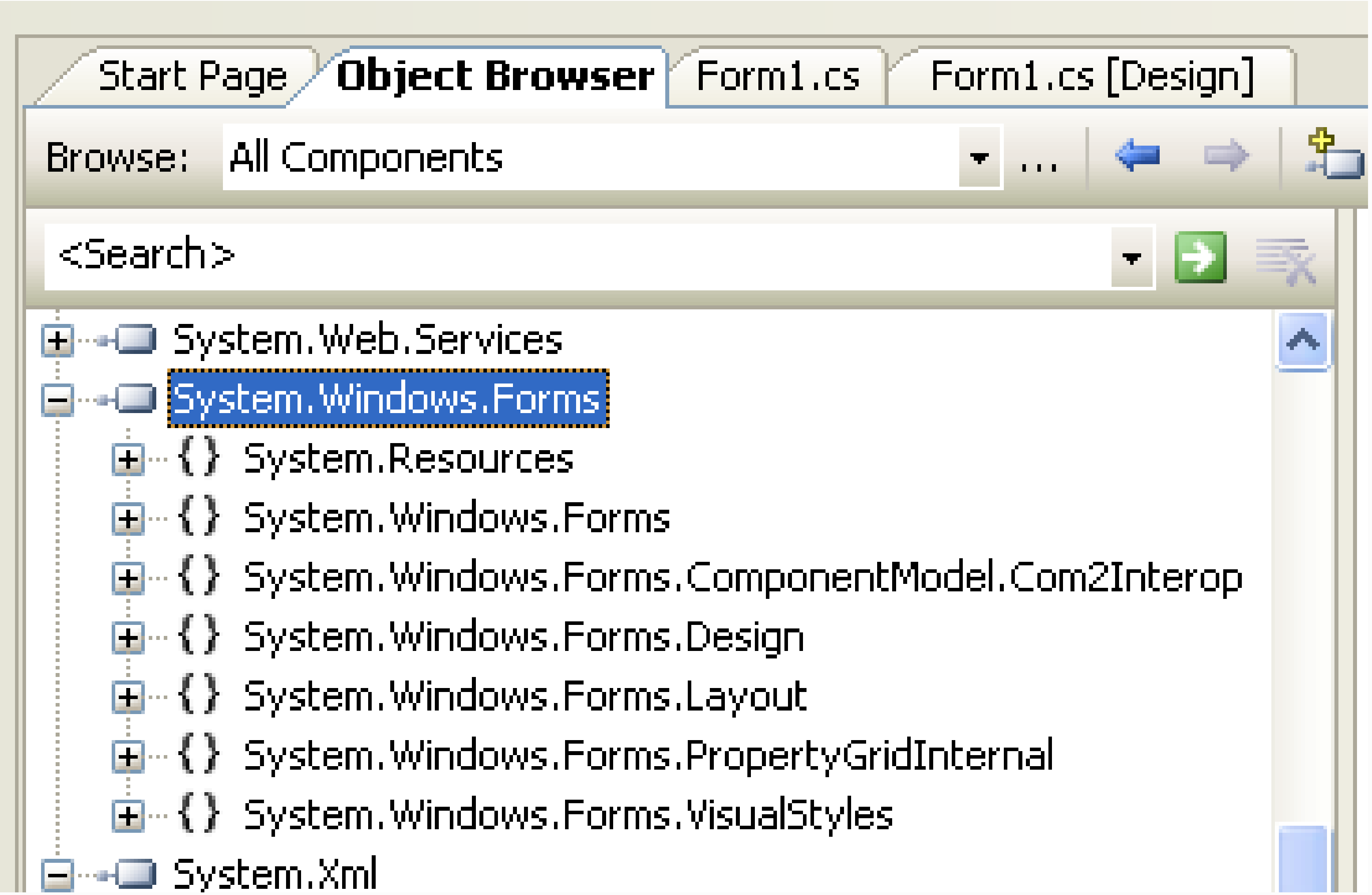
Objectives

- At the end of this session, you will be able to
  - Differentiate between SDI and MDI applications.
  - Create windows applications using Winforms.
  - Use Windows Form Control Collection to build highly interactive UI.
  - List common properties, methods and events of Control Class.
  - Use common dialog boxes provided in the library.
  - Create a custom dialog box.
  - Use GDI+ to draw graphical data programmatically.
  - Use Graphics class for graphics drawing

Windows Application

- Single Document Interface(SDI)
  - Opens a single document.
  - e.g. Notepad , WordPad
- Multiple Document Interface(MDI)
  - Multiple documents in a single application.
  - e.g. Microsoft Excel

Windows Forms Namespace



System.Web.Services
System.Windows.Forms
{ } System.Resources
{ } System.Windows.Forms
{ } System.Windows.Forms.ComponentModel.Com2Interop
{ } System.Windows.Forms.Design
{ } System.Windows.Forms.Layout
{ } System.Windows.Forms.PropertyGridInternal
{ } System.Windows.Forms.VisualStyles
System.Xml

System.Windows.Forms Namespace

- Classes in `System.Windows.Forms` Namespace
  - Core Infrastructure
    - Classes as Form, Application
  - Controls
    - Classes to create rich UI (Button, Menustrip,etc)
  - Components
    - Classes such as Tooltip, Errorprovider, etc.
  - Common Dialog Boxes
    - Classes such as OpenFileDialog, SaveAsDialog, ColorDialog, PrintDialog,etc.

First Windows Forms Application

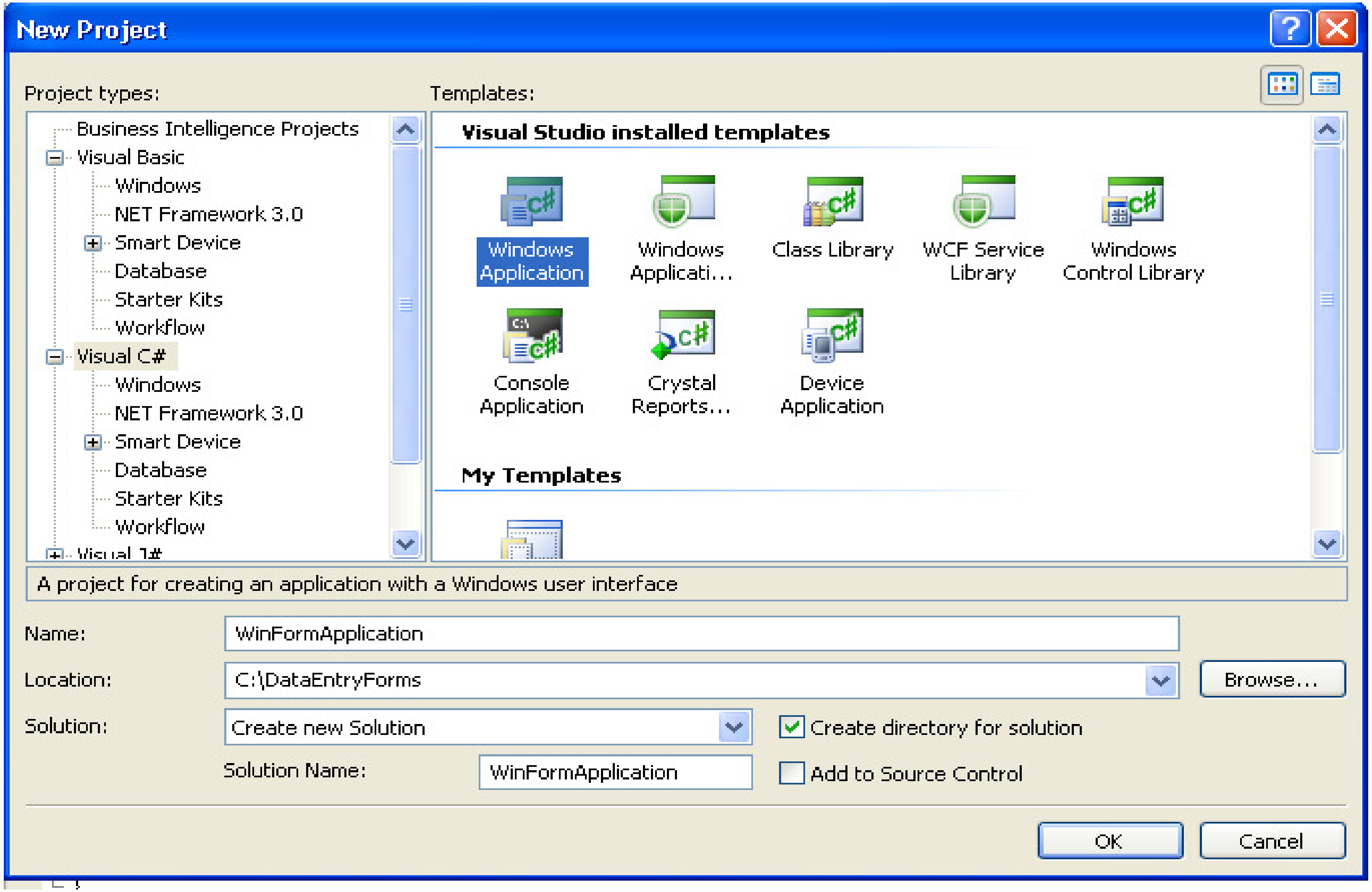
```
using System;
using system.Windows.Forms;
namspace SimpleApp
{
  class Program
  {
      static void Main()
      {
            Application.Run(new MainWindow());
      }
  }
}
class MainWindow:Form {}
}
```

Windows Form- Control Collection

```
class MainWindow:Form{
   private MenuStrip mnuMainMenu=new MenuStrip();
   private ToolStripMenuItem mnuFile=new ToolStripMenuItem();
   private ToolStripMenuItem munFileExit= new ToolStripMenuItem();
   public MainWindow(){  BuildMenuSystem();}

   private void BuildMenuSystem()     {
   mnuFile.Text="&File";
   mnuMainMenu.Items.Add(munFile);
   mnuFileExit.Text="E&xit);
   mnuFile.DropDownItems.Add(mnuFileExit);
   mnuFileExit.Click+=new
   System.EventHandler(this.mnuFileExit_Click);
   Controls.Add(this.mnuMainMenu);
   }
   private void mnuFileExit_Click(object sender, EventArgs e){
   Application.Exit();
   }
 }
```

Building GUI using Visual Studio.NET



New Project

Project types: Visual C#

Templates: Windows Application

A project for creating an application with a Windows user interface

Name: WinFormApplication

Location: C:\DataEntryForms

Solution: Create new Solution

Solution Name: WinFormApplication

☑ Create directory for solution

□ Add to Source Control

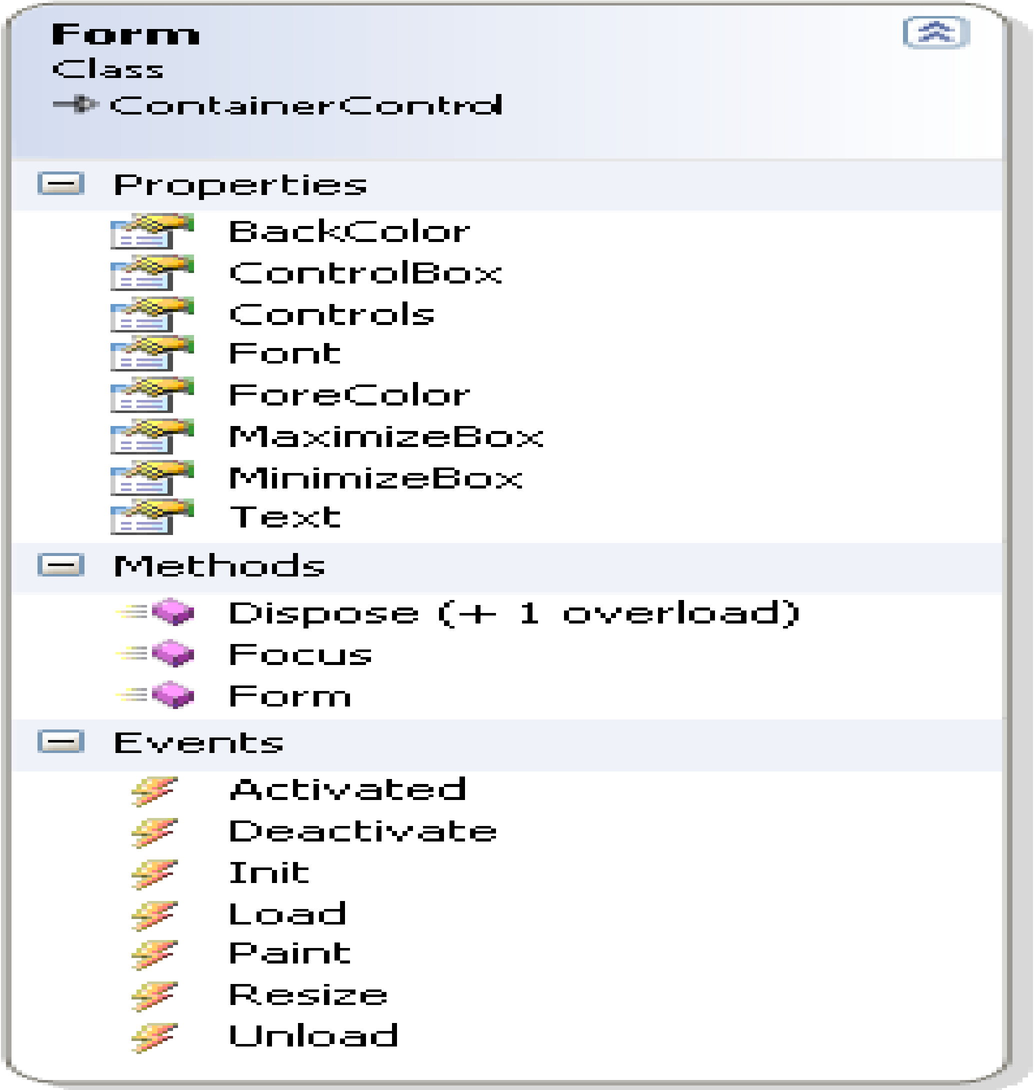
Form

- Represents an object of Form class present System.Windows.Forms namespace.
- From type
  - Represents Main Window, Child Window, MDI windows

```
partial class EmployeeForm
{
private void InitializeComponent(){
this.button2 = new System.Windows.Forms.Button();
this.button1 = new System.Windows.Forms.Button();
this.listBox1 = new System.Windows.Forms.ListBox();
this.label2 = new System.Windows.Forms.Label();
this.label1 = new System.Windows.Forms.Label();
this.textBox2 = new System.Windows.Forms.TextBox();
. . . //code
}
```

Code used by IDE to set the propertie s of controls to be used by main form

Common members of a `Form` Class

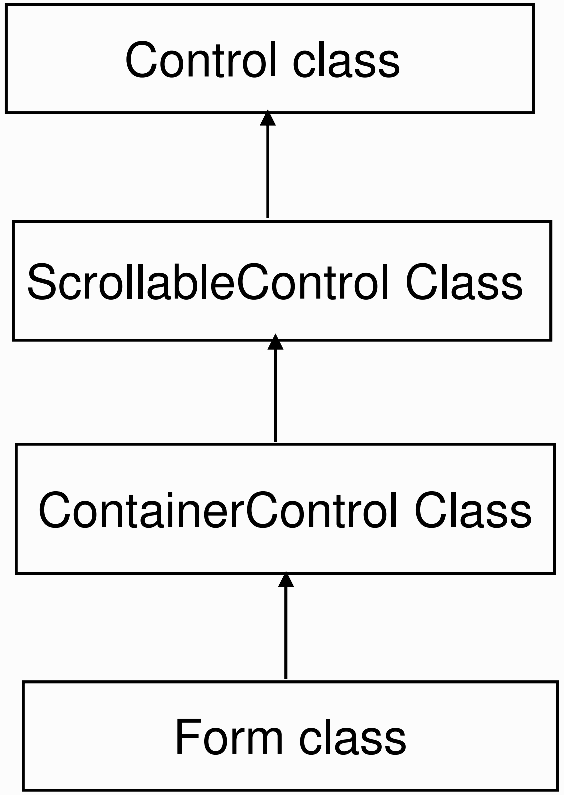


Form
Class
→ ContainerControl

Properties
- BackColor
- ControlBox
- Controls
- Font
- ForeColor
- MaximizeBox
- MinimizeBox
- Text

Methods
- Dispose (+ 1 overload)
- Focus
- Form

Events
- Activated
- Deactivate
- Init
- Load
- Paint
- Resize
- Unload



Control class

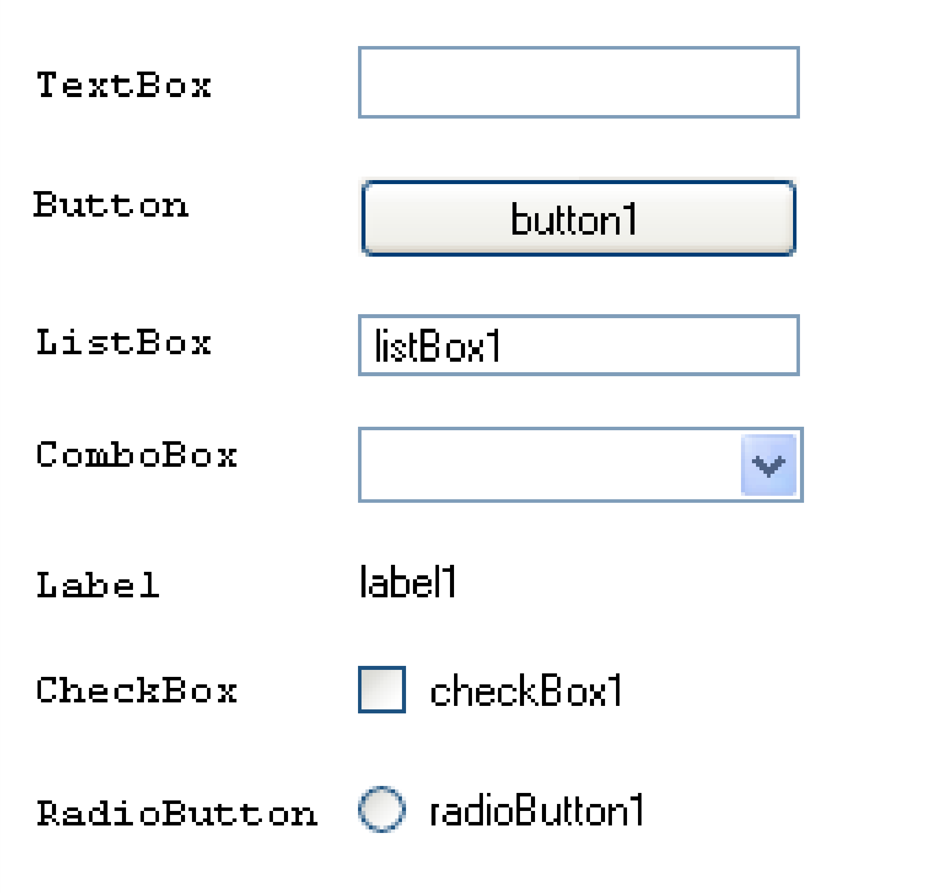
ScrollableControl Class

ContainerControl Class

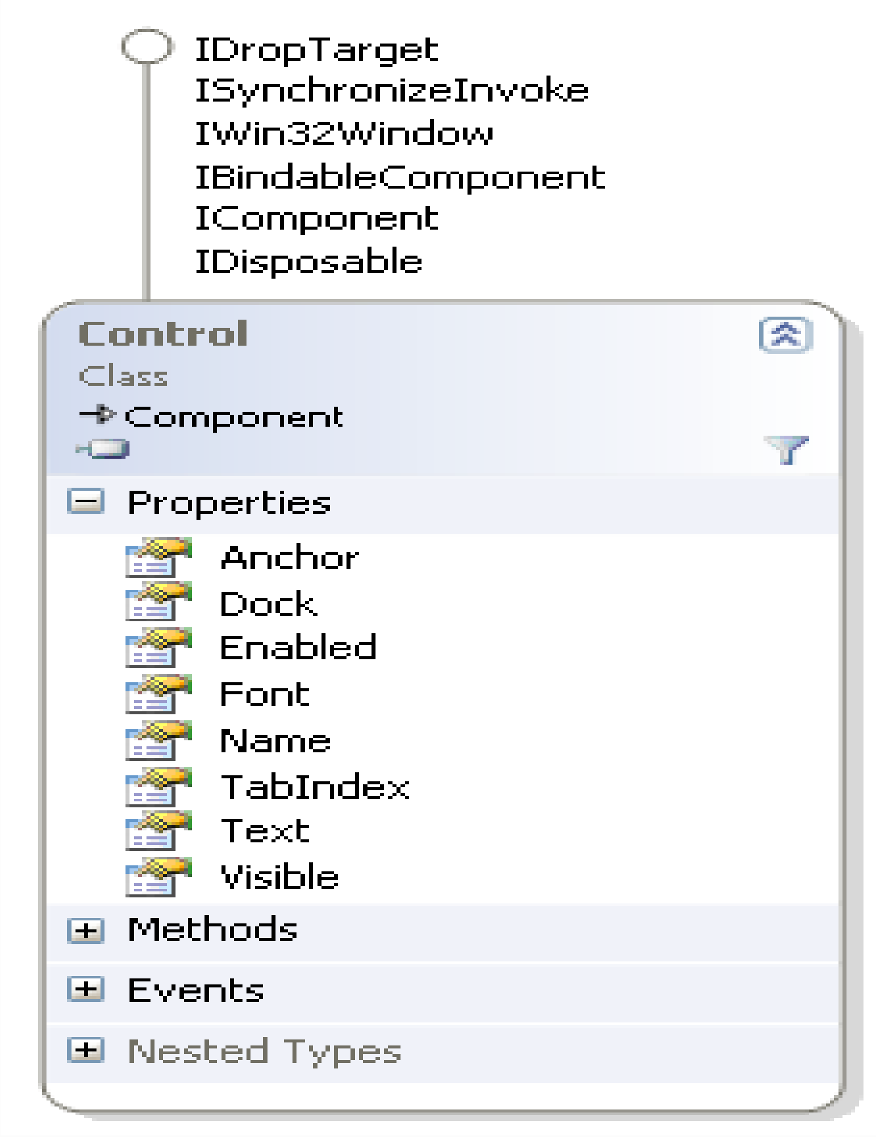
Form class

Controls

- `Control` class is the base class of all controls
- Common controls



Common properties & Events of `Control` Class



IDropTarget
ISynchronizeInvoke
IWin32Window
IBindableComponent
IComponent
IDisposable

Control
Class
→ Component

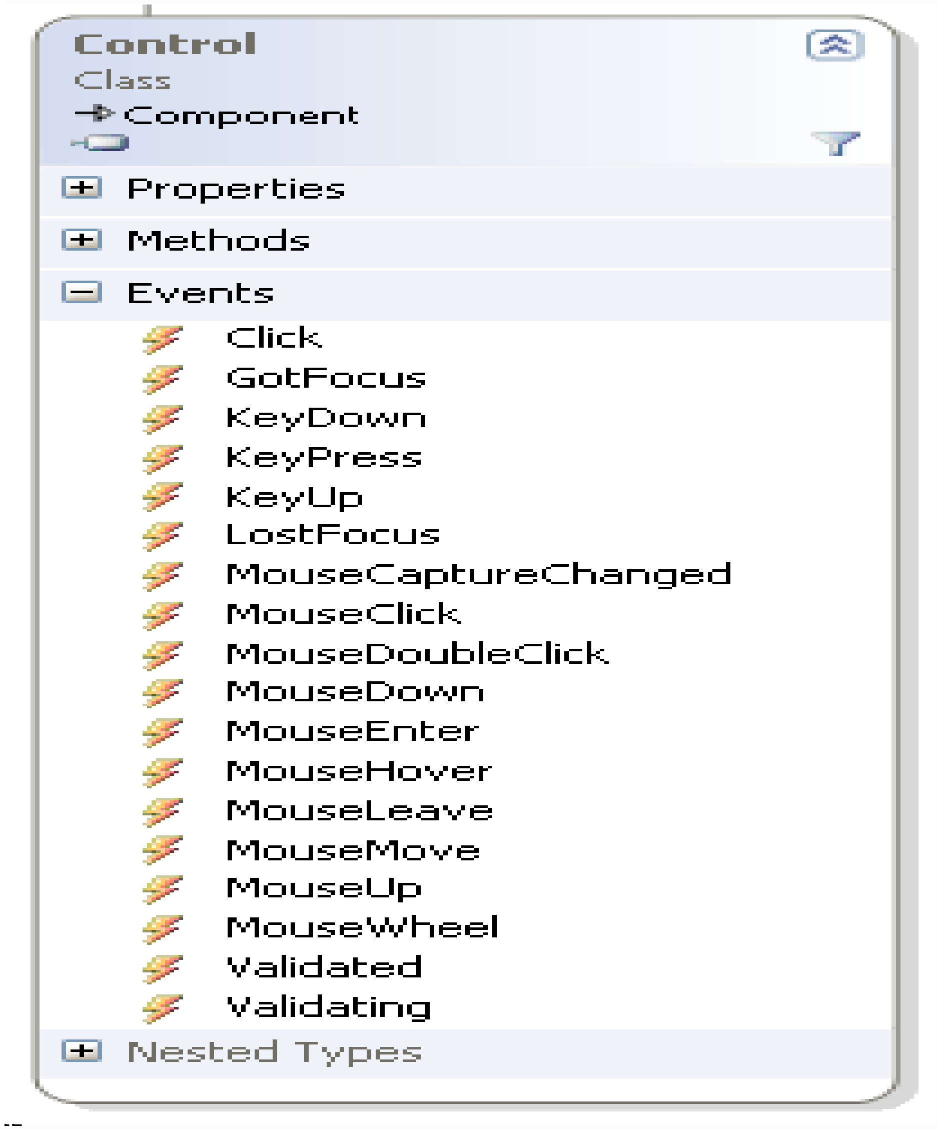
⊟ Properties

- Anchor
- Dock
- Enabled
- Font
- Name
- TabIndex
- Text
- Visible

⊞ Methods

⊞ Events

⊞ Nested Types



Control
Class
→ Component

⊞ Properties

⊞ Methods

⊟ Events

- Click
- GotFocus
- KeyDown
- KeyPress
- KeyUp
- LostFocus
- MouseCaptureChanged
- MouseClick
- MouseDoubleClick
- MouseDown
- MouseEnter
- MouseHover
- MouseLeave
- MouseMove
- MouseUp
- MouseWheel
- Validated
- Validating

⊞ Nested Types

Mouse and Keyboard Event Handling

- Set of Events for monitoring mouse and Keyboard events.

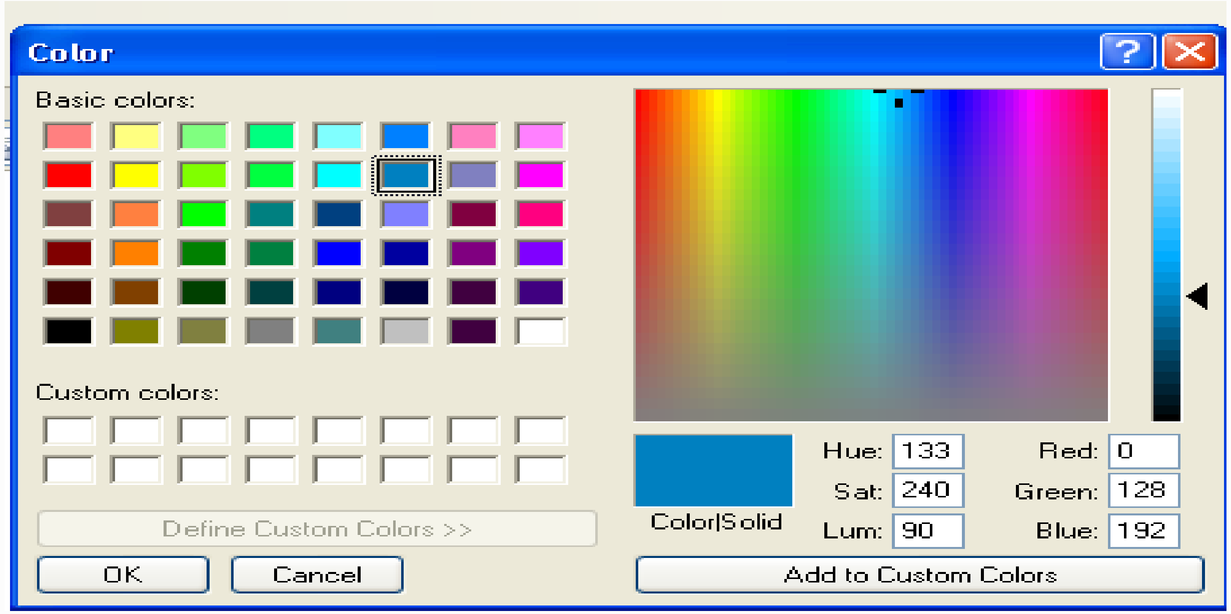
```
private void MainWindow_MouseMove(object sender,
MouseEventArgs e)
{
Text=string.Fromat(“Mouse Position:{0}”, e.Location);
}

private void MainWindow_KeyUp(object sender,
KeyEventArgs e)
{
Text=string.Fromat(“Key{0}”, e.KeyCode.ToString());
}
```

Dialog boxes

- Use for data capturing from user in Application.
- No base class named Dialog
- `System.Windows.Form` as base class
- Winforms provide common dialog classes
  - `OpenFileDialog`
  - `PrintDialog`
  - `ColorDialog`
  - `FontDialog`
  - etc.

Using ColorDialog

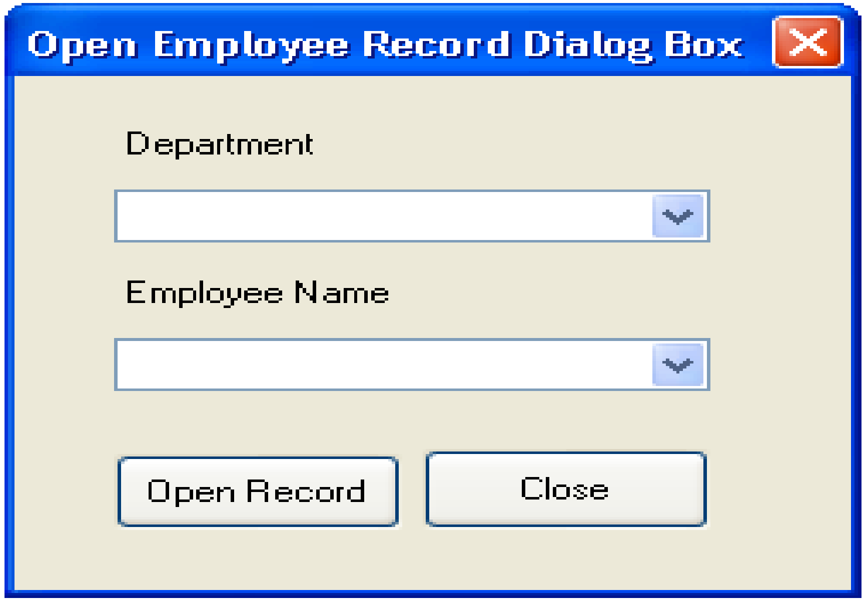


Hue: 133
Sat: 240
Lum: 90
Red: 0
Green: 128
Blue: 192

```
Private void MainWindow_Oncolor(object sender,
MouseEventArgs e) {
          ColorDialog dlg=new ColorDialog();
          if((dlg.ShowDialog())==DialogResult.OK)
          {
                    Color c = dlg.Color;
           }
}
```

Creating Custom Dialog Boxes

- Create windows form using template **(Visual Studio. Net)**
- Design and Set following properties of Form



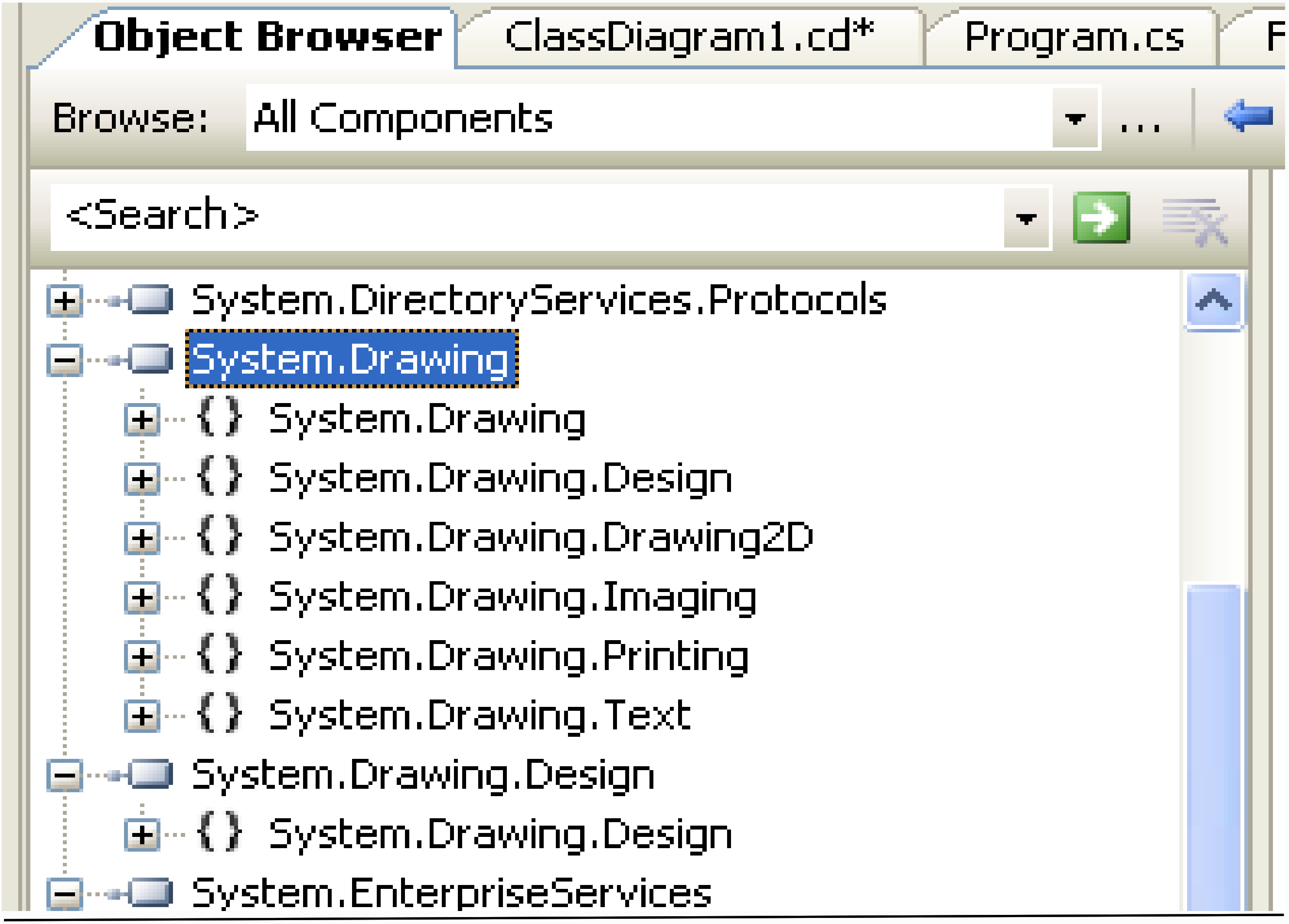
Property	**Set value**
`FormBorderStyle`	FixedDialog
`MinimizeBox`	False
`MaximizeBox`	False
`StartPosition`	CenterParent
`ShowInTaskbar`	false

- Crate instance of Custom dialog class.
- Invoke `ShowDialog`() method

GDI +

- Provides ability to draw graphical data programmatically on Form.
  - i.e. drawing Pie char to display items in stock.
- GDI + support provided by classlibrary
  - `System.Drawing.dll`

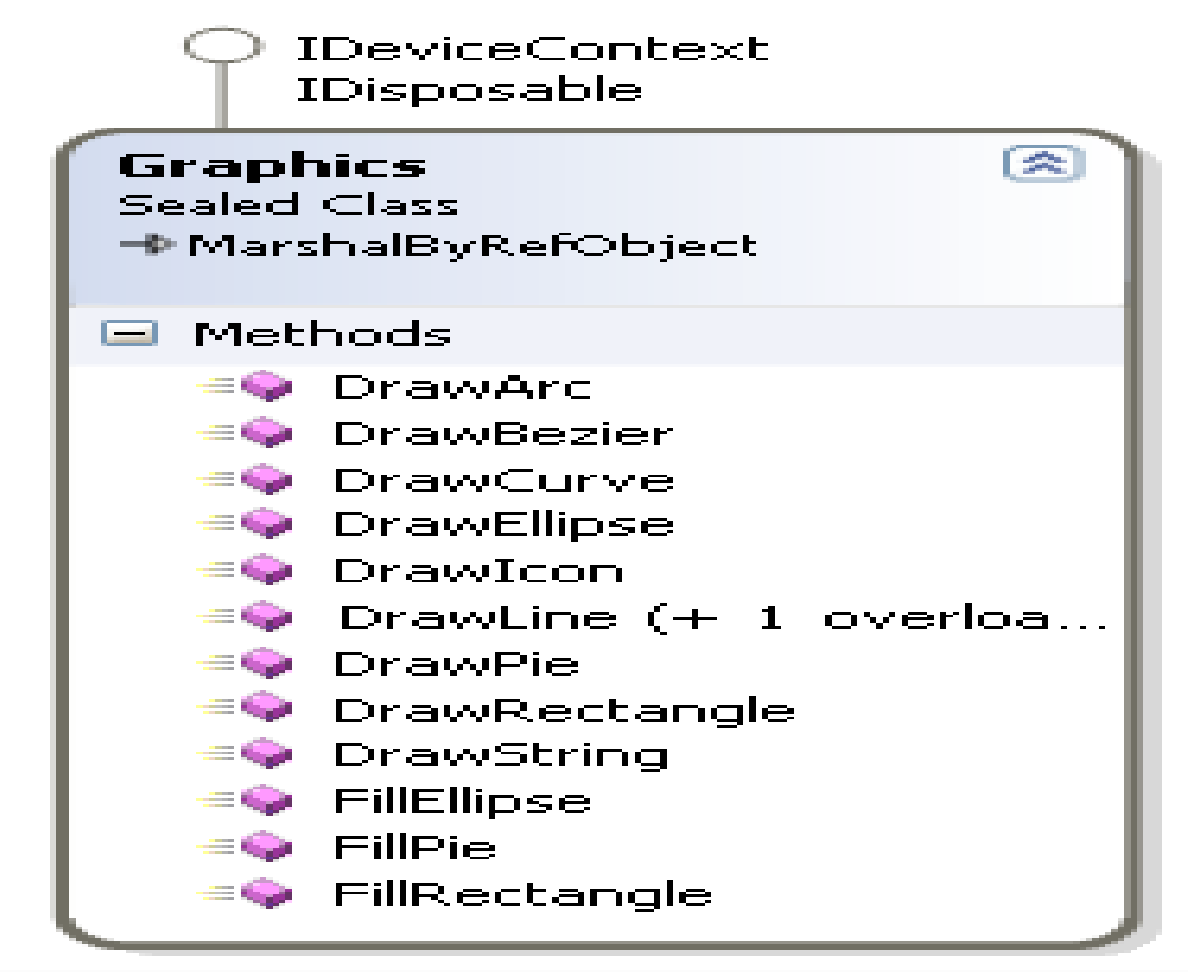
GDI + Namespaces



System.DirectoryServices.Protocols
System.Drawing
{ } System.Drawing
{ } System.Drawing.Design
{ } System.Drawing.Drawing2D
{ } System.Drawing.Imaging
{ } System.Drawing.Printing
{ } System.Drawing.Text
System.Drawing.Design
{ } System.Drawing.Design
System.EnterpriseServices

Graphics Class

- Core class for Graphics Drawing
- Device context of surface to be drawn



IDeviceContext
IDisposable

Graphics
Sealed Class
MarshalByRefObject

Methods

DrawArc
DrawBezier
DrawCurve
DrawEllipse
DrawIcon
DrawLine (+ 1 overloa...
DrawPie
DrawRectangle
DrawString
FillEllipse
FillPie
FillRectangle

Handling Paint Event

```
Private void MainWindow_Paint(object
sender,PaintEventArgs e)
{
      Graphics g=e.Graphics;
      g.DrawLine(Pens.Red, 10,10,300,300);
      g.DrawString(“Grahphics”, new Font(“Arial”,30));
      Using (Pen p=new Pen(Color.Yellow,10))
      {
            g.DrawLine(p,20,50,300,300)
      }

}
```

Quick Recap …

- Windows applications are either SDI Applications or MDI Application.
- Winforms is an object model which consists of number of classes, structures, enums and namespaces for creating SDI and MDI applications.
- Winforms functionality is provided by `System.Windows.Forms.dll`.
- We can add child controls such as Button, Textbox, and Menustrip to provide rich user interface for Form.
- MainWindow, ChildWinows are created using Form type .
- Control class is the base class for controls.
- Mouse event worked in conjunction `MouseEventHandler` delegate.
- Users input are captured in typical windows application using Dialog box.
- Graphical rendering is done in windows applications using GDI +.
- Whenever window is resized, framework fires paint event of window