Instructor: Bruce Reynolds

Introduction to Applications in C# Class 6

Concepts from Last Week

- Classes, structures, and enums
- Methods
  - Constructors
  - ToString

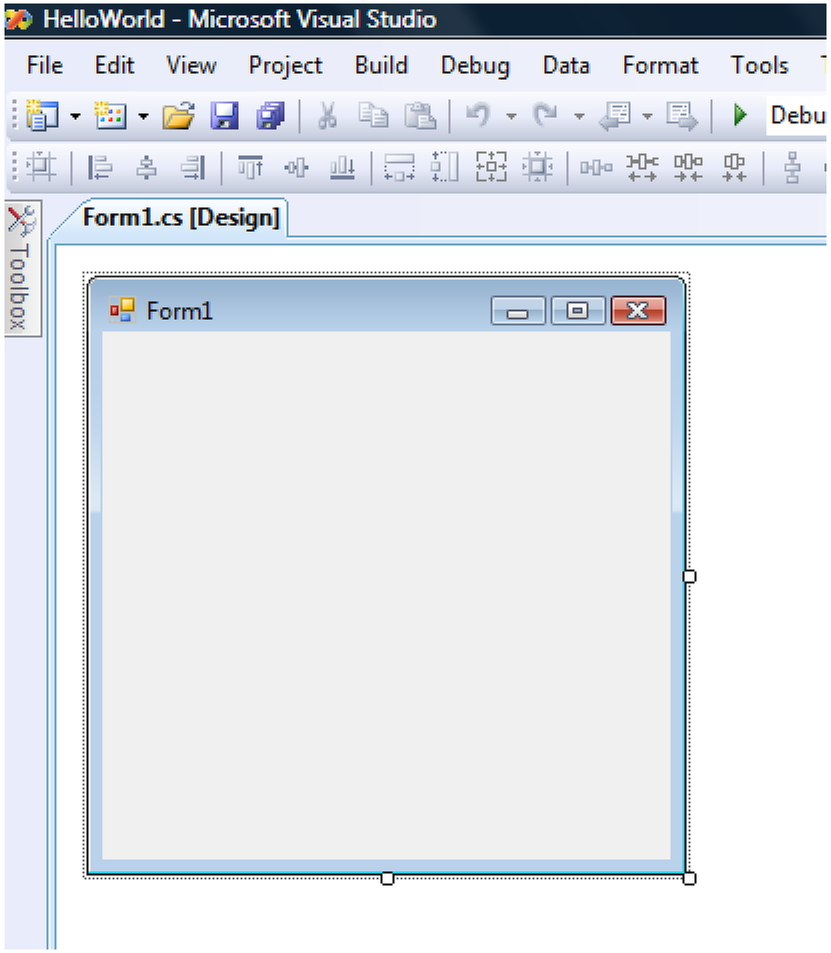
Homework Review

Concepts for Week 6

- Window Forms
  - Windows Forms designer
  - Event-driven programming
  - Controls
  - Data conversion
  - A note about WPF

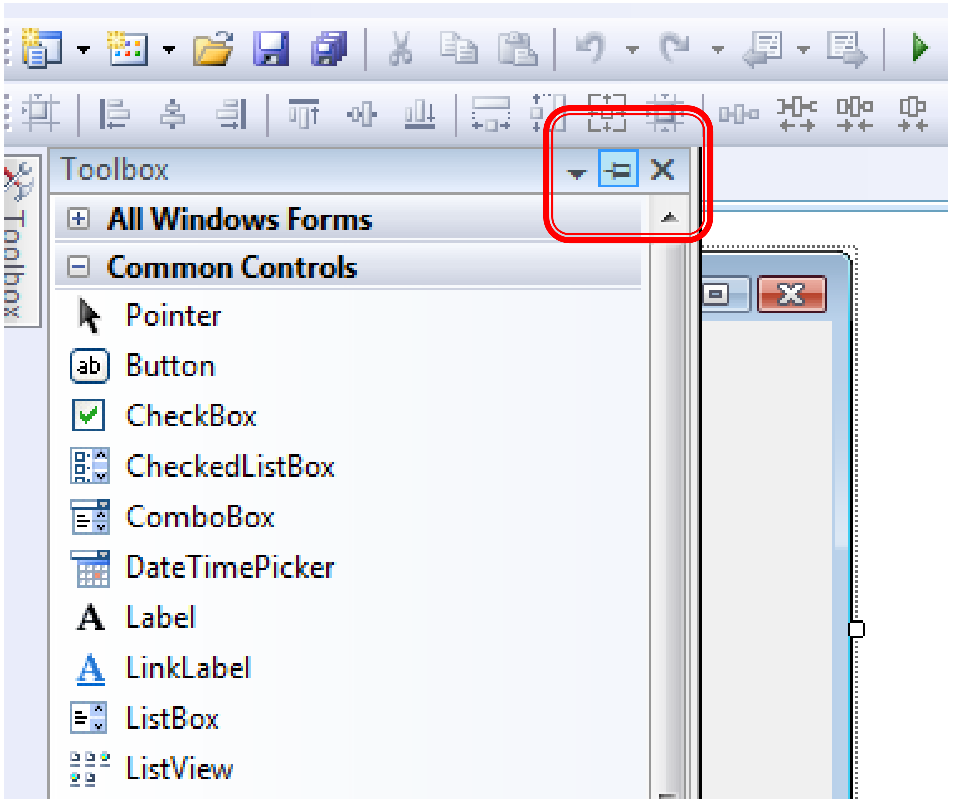
Hello World

- Create a new Windows project:
  - File | New | Project
  - Visual C# | Windows | Windows Forms Application
  - Name the project "HelloWorld".



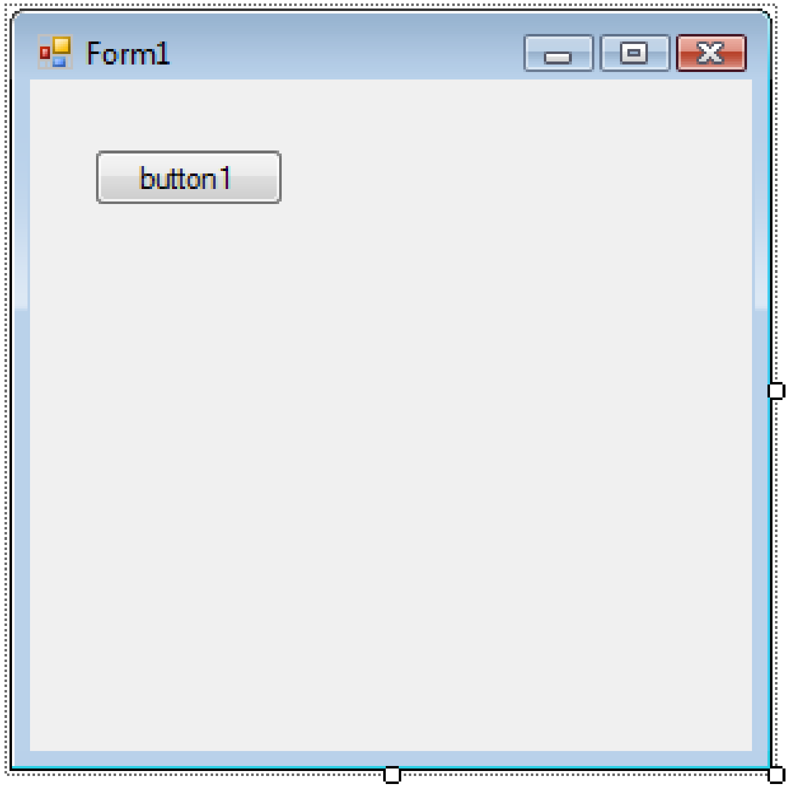
Hello World, continued

- Open the Toolbox and pin it open.
- Open the Common Controls section.



Hello World, continued

- Drag a button from the Toolbox onto the form.



Hello World

- Double-click the button.
- In the code editor add this code to the button1_Click method:

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Hello World");
}
```

- Press F5 to run the program.
- Click the button.

Windows Forms Designer

- The designer lets you design the user interface.
  - This is the "design-time" experience.
  - When you press F5, you are in "run-time."
- The Toolbox has controls (UI elements) that you can drag onto the form and configure:
  - Properties window- the grid in the lower right
  - Smart tags – the little arrow next to some controls
  - Try it!

Event-Driven Programming

- Controls **raise events** that you can **handle** with methods.
  - The method must have a particular **signature**.
  - The event is hooked up to the control in designer-generated code. Do not add this code yourself.
- Events are class members.
  - In this class, we write code that **subscribes** to events.
  - Next quarter, you'll add events to classes that you write.

Event-Driven Programming

- Windows forms applications are controlled by responding to events generated when the user interacts with the UI.
- Each control has a default event.
  - Double-click on a control to learn its default event. That gives you a clue about what the control is used for.
- You can choose to respond to an event or not.

Method(object sender, EventArgs e)

- All event methods follow this general signature.
- “sender” is the object that generated the event.
- “e” is a parameter that carries more information about the event.
  - Today’s examples don’t demonstrate this.

Echo Text

- Create a new project, EchoText.
- Drag a TextBox onto the form.
- Drag a Button onto the form.

Echo Text

- Double-click the button.
- In the code editor add this code to the button1_Click method:

```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show(textBox1.Text);
}
```

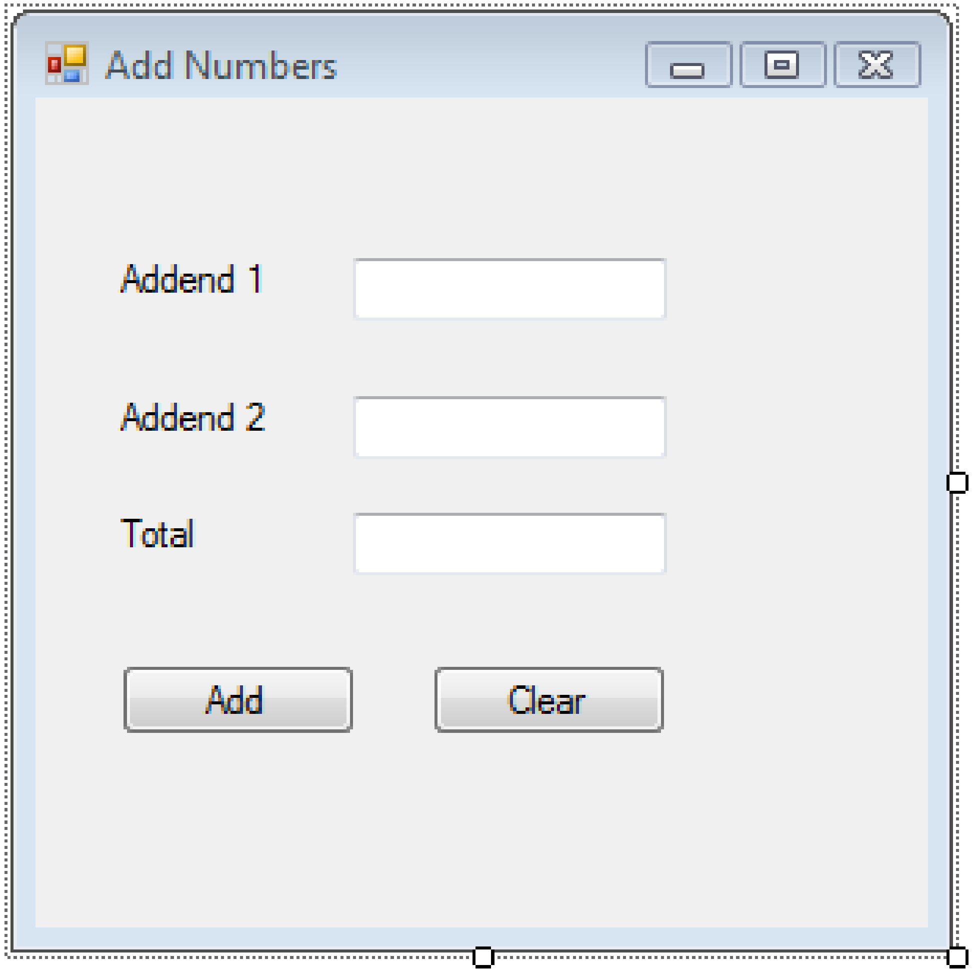
- Controls have properties to access their data.
  - You can set the properties in the Properties window. Try it!
  - You can set the properties in Code.

Set Properties at Run-Time

- Change the code in button1_Click to:

```
private void button1_Click(object sender, EventArgs e)
{
    button1.BackColor = Color.Orange;
}
```

Simple Addition



Simple Addition

- Create a new project named SimpleAddition.
- Add three MaskedTextBox controls.
  - Name – ctlAddend1, ctlAddend2, ctlTotal
- Add three Label controls.
- Add two Button controls.
  - Name – ctlAdd, Text – Add
  - Name – ctlClear, Text - Clear
- Form (the form has properties, too)
  - Text – Add Numbers

Simple Addition

- Configure the MaskedTextBox controls:
  - Click the SmartTag.
  - Choose Set Mask.
  - Choose “Numeric (5 digits)”

Add a Clear Button

- Double-click the Clear button and add code :

```
private void ctlClear_Click(object sender, EventArgs e)
{
    ctlAddend1.Text = "";
    ctlAddend2.Text = "";
    ctlTotal.Text = "";
}
```

Simple Addition

- Double-click the Add button and add code:

```
private void ctlAdd_Click(object sender, EventArgs e)
{
    int add1 = int.Parse(ctlAddend1.Text);
    int add2 = int.Parse(ctlAddend2.Text);
    int sum = add1 + add2;
    ctlTotal.Text= sum.ToString();
}
```

- F5 to run.

The Form is a Class

- Notice this code in the code editor

```
public partial class Form1 : Form
```

- The form is a class that inherits from System.Windows.Form.
- All the controls on the form are class members (fields) of the Form1 class.
  - Their declaration and initialization is hidden in the designer generated code in Form1.Designer.cs. Don't mess with the .Designer.cs file.
- The controls are instances of other objects (TextBox, Label, Button, etc.)

Simple Addition – Version 2

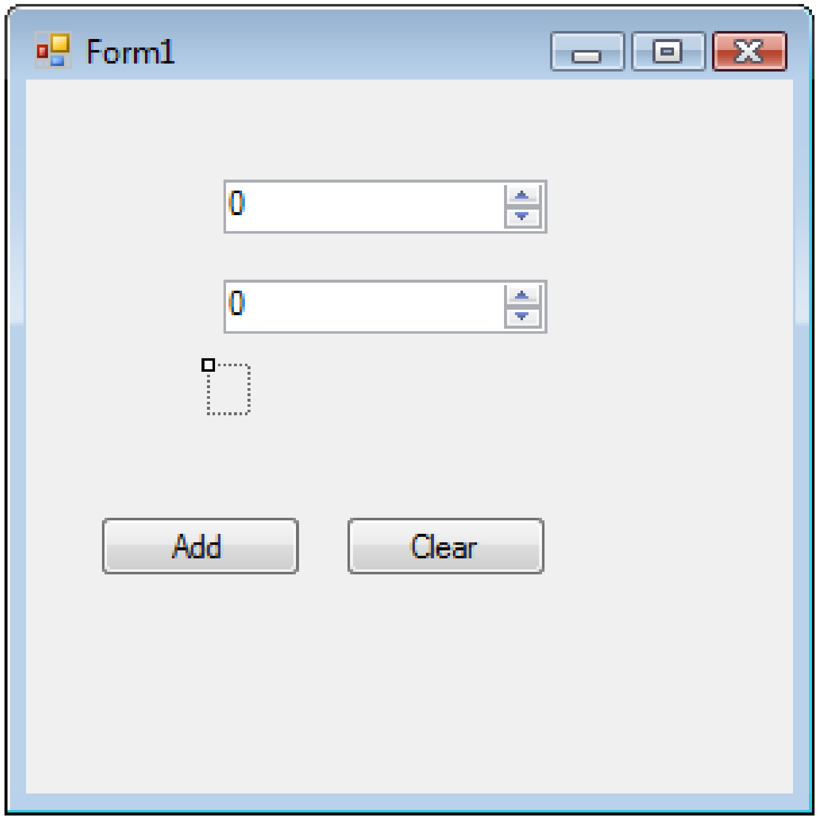
- That's great, but we're still doing int.Parse.
  - Requires the user to be careful about input.
  - Requires the developer to write error code.
  - And we really don't want the user to be able to type in the result.

Simple Addition – Version 2

- Create a new project, BetterAddition.
- Add these controls and set the properties:

Control	Name	Text
NumericUpDown	ctlAddend1	
NumericUpDown	ctlAddend2	
Label	ctlTotal	“”
Button	ctlAdd	Add
Button	ctlClear	Clear

Simple Addition – Version 2

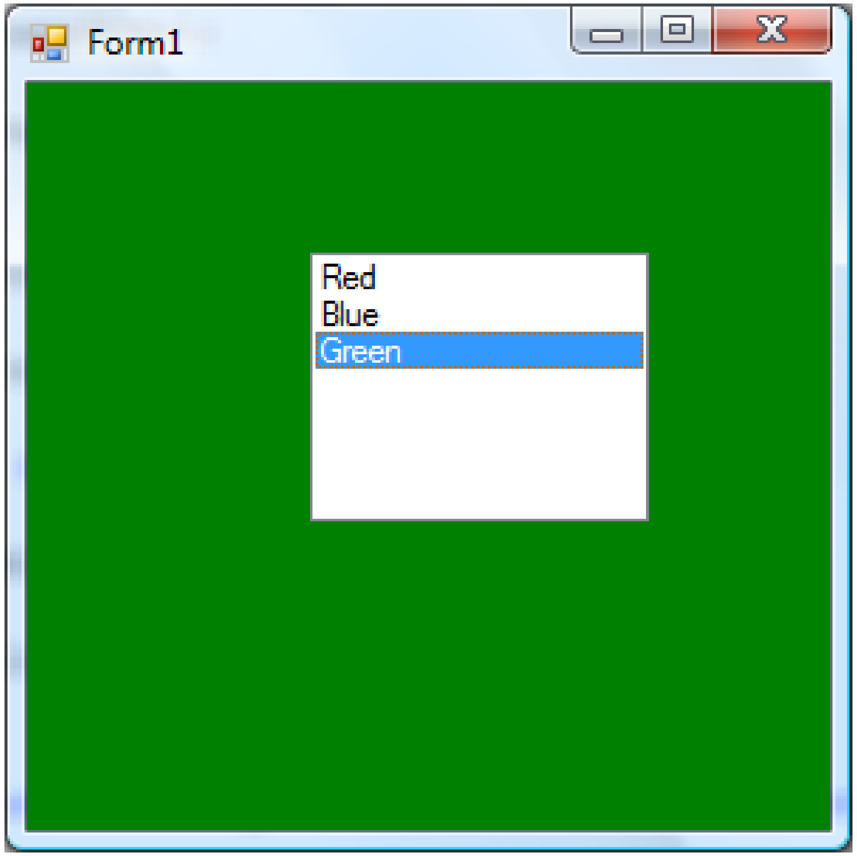


Simple Addition – Version 2

- Add this code to the click events:

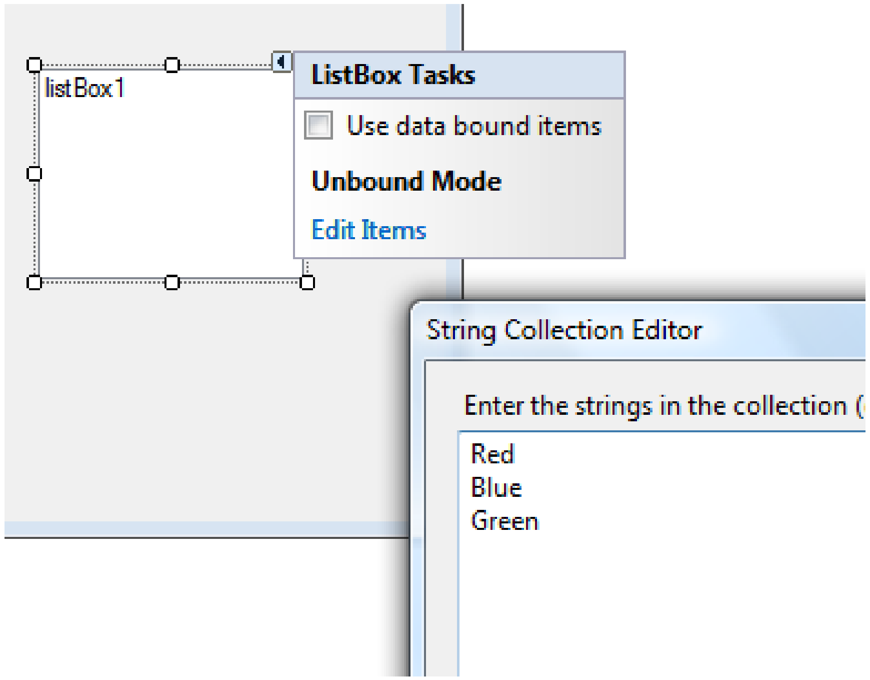
```
private void ctlAdd_Click(object sender, EventArgs e)
{
    decimal add1 = ctlAddend1.Value;
    decimal add2 = ctlAddend2.Value;
    decimal sum = add1 + add2;
    ctlTotal.Text = sum.ToString();
}

private void ctlClear_Click(object sender, EventArgs e)
{
    ctlAddend1.Value = 0;
    ctlAddend2.Value = 0;
    ctlTotal.Text = "";
}
```

Color Preferences



Color Preferences

- Create a new Windows Application project, MyFavoriteColors.
- Add a ListBox control to the form.
- Using the SmartTag, Edit Items, add these values:
  - Red, Blue, and Green



Enter the strings in the collection (

Red
Blue
Green

Color Preferences

- Double-click the ListBox and add this code:

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    switch (listBox1.SelectedItem.ToString())
    {
        case "Red" :
            this.BackColor = Color.Red;
            break;
        case "Blue" :
            this.BackColor = Color.Blue;
            break;
        case "Green" :
            this.BackColor = Color.Green;
            break;
    }
}
```

Conversions

- A conversion lets you treat data of one type as data of another type.
  - For example, we've converted strings to ints.
- Conversions are either:
  - Implicit – You don't need to do anything to make these work.
    - Int -> decimal
    - Anything -> Object
  - Explicit
    - You have to use a casting expression or a conversion method.
    - The conversion may fail if the conversion isn't defined, or the data can't be converted.

Conversion Examples

- Implicit

```
int a = 123;
long b = a;
```

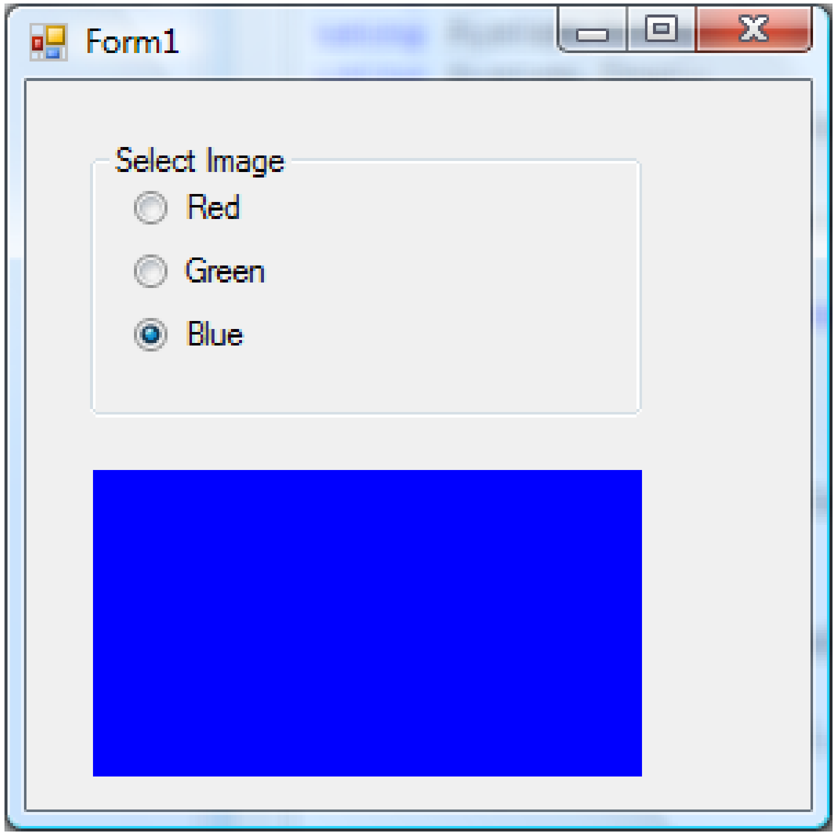
- Explicit

```
decimal d = 234;
int e = (int)d;
int f = int.Parse(“234”);
```

Boxing – Conversion to Object

- The Items collection of the ListBox control holds Object instances.
  - If you put something else in, like a string ("Green"), the runtime **boxes** it into an Object instance.
  - When you retrieve the items from the list box, they are considered Object.
    - You need to convert them back to the type that you put it, using a cast expression.
- In the color example, we converted the Item to string and did a comparison. (Or we could have used listBox1.Text and made it simpler. ☺)

RadioButton and PictureBox



Red [✓ Blue] Green

RadioButton and PictureBox

- Create a new project, PictureChooser.
- Add a GroupBox control. It is part of the Containers group in the Toolbox.
- Add three RadioButton controls to the GroupBox.
- Add a PictureBox control.
- Set the properties as shown on the next slide.

RadioButton and PictureBox

Control	Property	Value
GroupBox	Text	Select Image
RadioButton1	Name	ctlRed
	Text	Red
RadioButton2	Name	ctlGreen
	Text	Green
RadioButton3	Name	ctlBlue
	Text	Blue
PictureBox	Name	pictureBox1
	BackColor	Blue

RadioButton and PictureBox

- Double-click each control to create an event handler.

RadioButton and PictureBox

- Add this code to the event handlers:

```
private void ctlGreen_CheckedChanged(object sender, EventArgs e)
{
    pictureBox1.BackColor = Color.Green;
}

private void ctlBlue_CheckedChanged(object sender, EventArgs e)
{
    pictureBox1.BackColor = Color.Blue;
}

private void ctlRed_CheckedChanged(object sender, EventArgs e)
{
    pictureBox1.BackColor = Color.Red;
}
```

A Note about WPF

- WPF
  - Windows Presentation Foundation
    - A newer framework that replaces Windows Forms, first version released in November 2006
    - Incorporates graphics, animation, audio & video
    - Uses XAML, a descriptive markup language used to define and arrange GUI controls
  - MSDN: http://msdn.microsoft.com/en-us/library/aa970268.aspx

Reading 6

- Deitel & Deitel
  - Chapter 14 – GUI with Windows Forms, Part 1
  - Chapter 15 – GUI with Windows Forms, Part 2
- MSDN: Windows Forms (http://msdn.microsoft.com/en-us/library/dd30h2yb.aspx)

Assignment 6