Radio Buttons and Check Boxes

GUIs commonly use radio buttons and check boxes to let the user select items.

Radio Buttons

Radio buttons are useful when you want the user to select one choice from several possible choices. At run time, only one radio button in a group may be selected at a time. Clicking on a radio button selects it and automatically deselects any other radio button in the same group. Figure 4-1 shows a form, at run time, with a group of three radio buttons.

Figure 4-1 Radio buttons

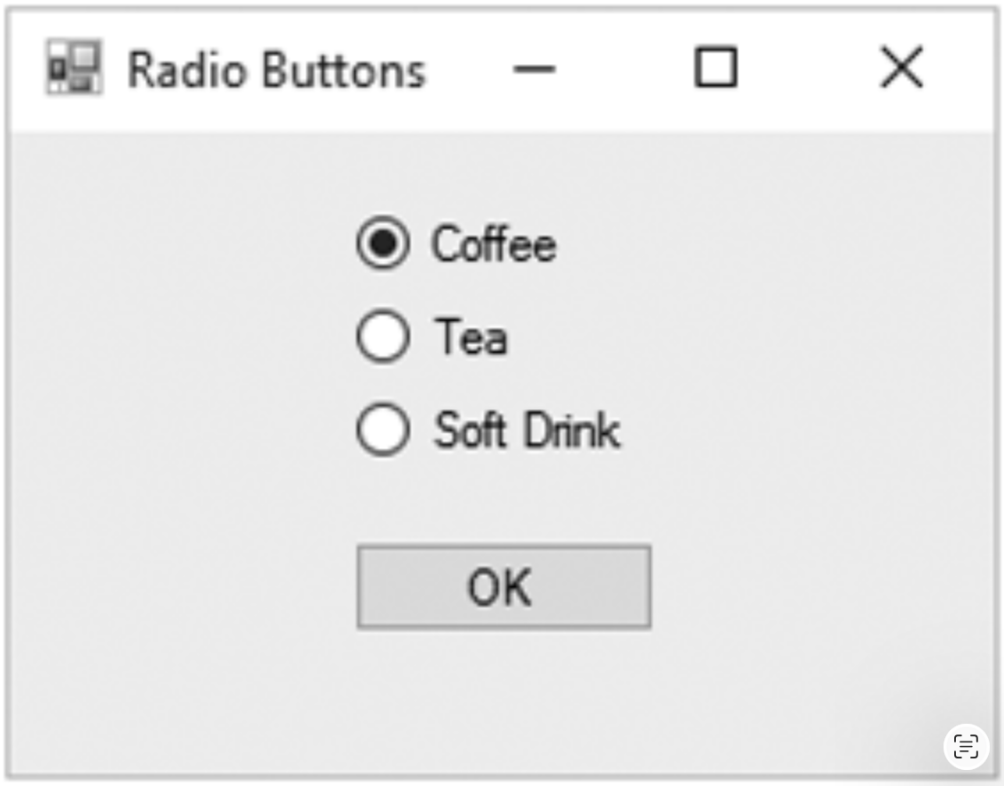
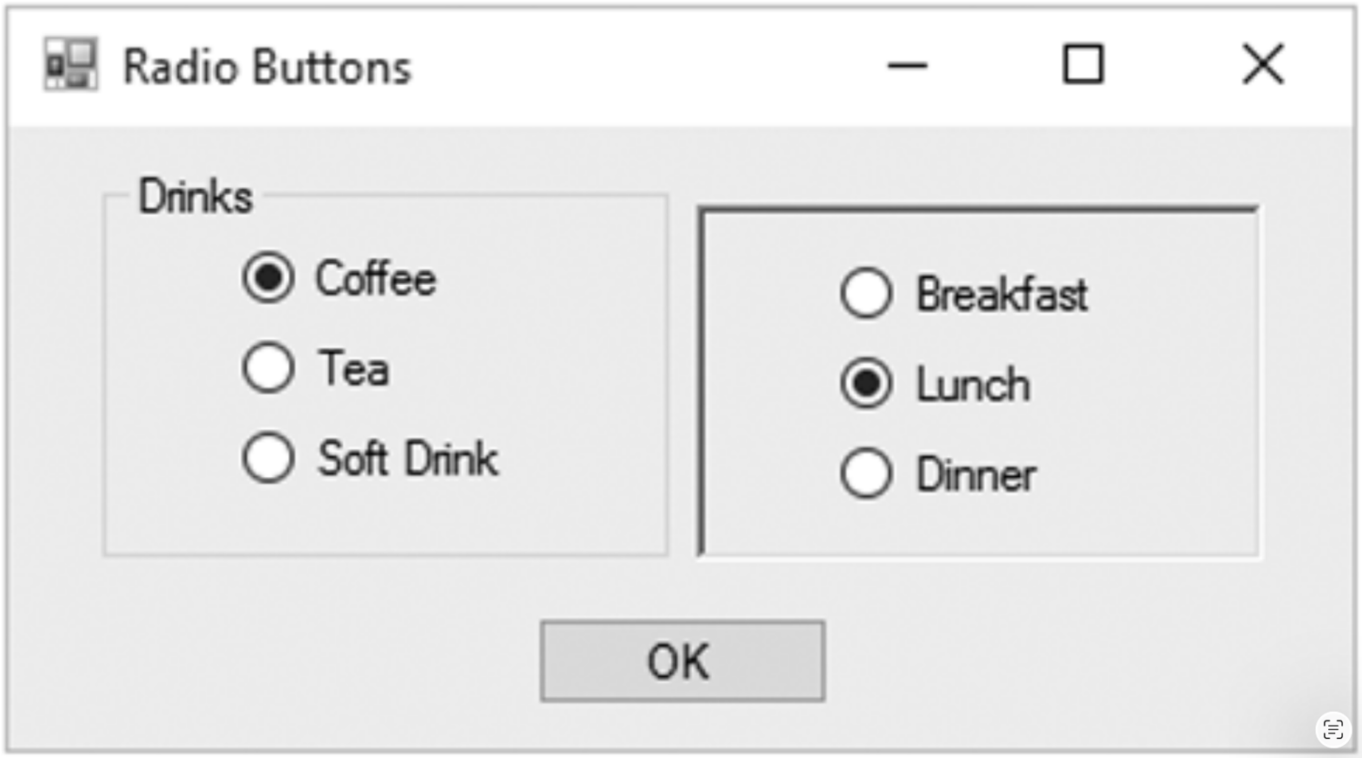


Figure 4-2 shows a form, where the group on the left is inside a GroupBox control, and the group on the right is inside a Panel control. When the application runs, the user will be able to **select only one RadioButton from each group**. In the figure, Coffee is selected in the left group and Lunch is selected in the right group.

Figure 4-2 A form with two groups of RadioButton controls



RadioButton controls have a **Text** property, which holds the text that is displayed next to the radio button's circle.

RadioButton controls have a **Checked** property that determines whether the control is selected or deselected. The Checked property is a Boolean property, which means that it may be set to either True or False.

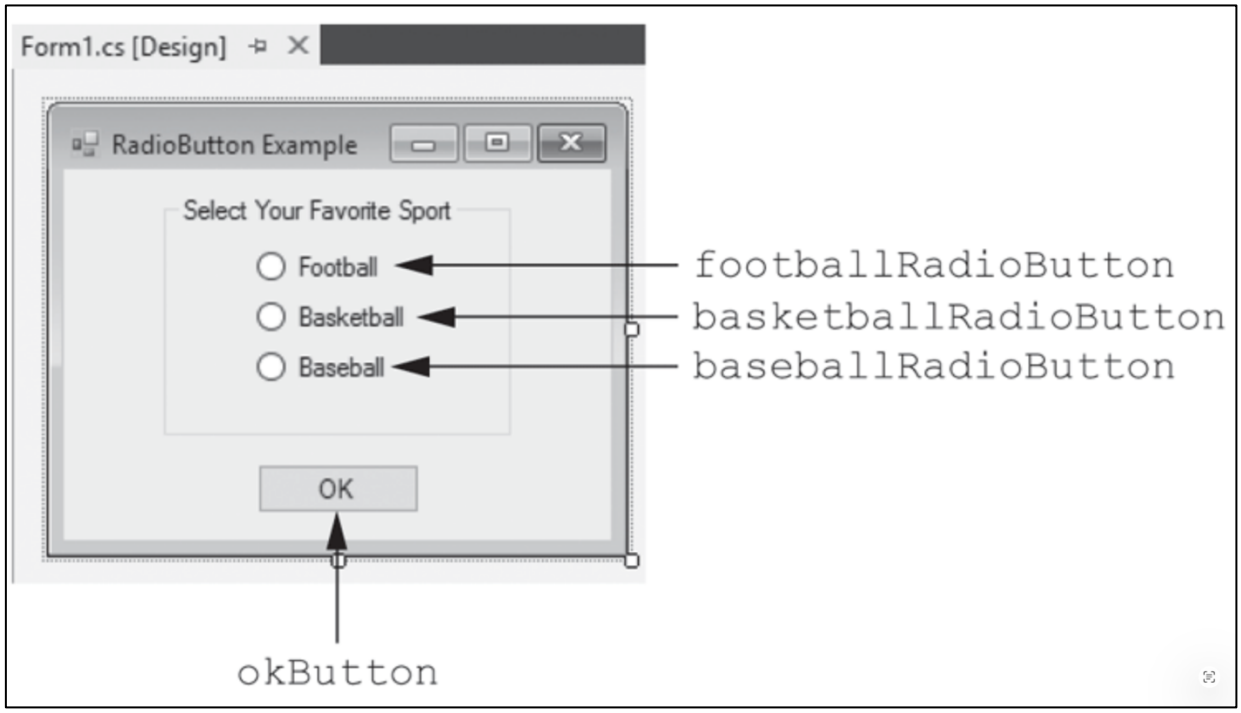
Working with Radio Buttons in Code

In code, you can determine whether a RadioButton control is selected by testing its Checked property. For example, suppose a form has a RadioButton control named choice1RadioButton. The following if statement determines whether it is selected:

```
if (choice1RadioButton.Checked == true)
{
   MessageBox.Show("You selected Choice 1.");
}
```

The application's form is shown in Figure 4-3. When you run the application, select one of the radio buttons and then click the OK button. A message box appears showing the sport that you selected.

Figure 4-3 The RadioButton Example form



Here is the code for the okButton_Click event handler:

```
1   private void okButton_Click(object sender, EventArgs e)
2   {
3       if ( footballRadioButton.Checked )
4       {
5           MessageBox.Show("You selected Football.");
6       }
7       else if ( basketballRadioButton.Checked )
8       {
9           MessageBox.Show("You selected Basketball.");
10      }
11      else if ( baseballRadioButton.Checked )
12      {
13          MessageBox.Show("You selected Baseball.");
14      }
15  }
```

Check Boxes

A check box appears as a small box with some accompanying (مصاحب) text. Figure 4-4 shows an example. They are called check boxes because clicking on an empty check box causes a check mark ✓ to appear in it. If a check mark already appears in a check box, clicking it removes the check mark.

Figure 4-4 A check box

Check boxes are similar to radio buttons, except that check boxes operate independently, allowing multiple selections at the same time. You can have one or more check boxes in a group, and any number of them can be selected at any given time.

CheckBox controls have a **Text** property, which holds the text that is displayed next to the check box.

CheckBox controls have a **Checked** property. When a CheckBox control is selected, or checked, its Checked property is set to True.

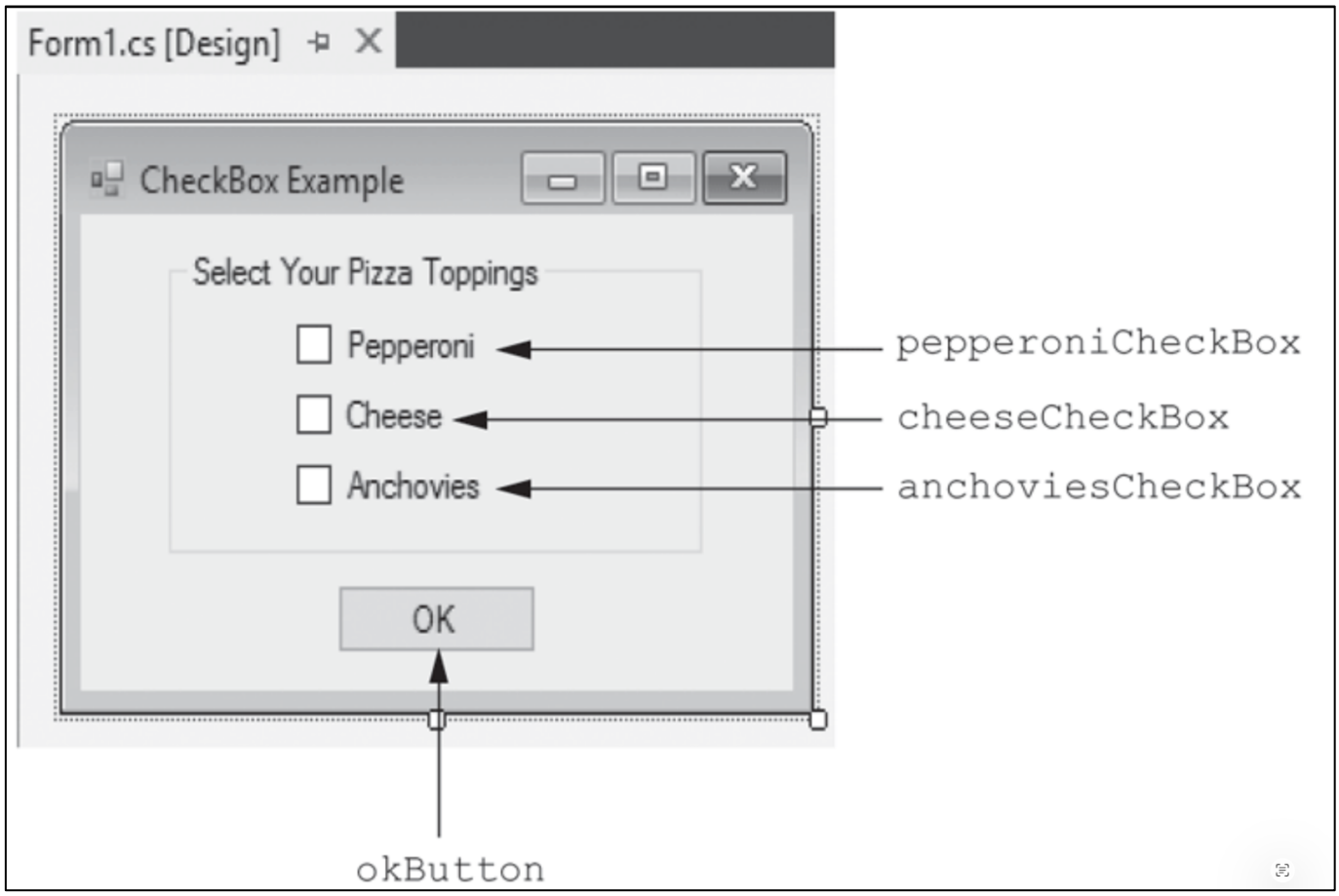
Working with CheckBox Controls in Code

In code, you can determine whether a CheckBox control is selected by testing its Checked property. For example, suppose a form has a CheckBox control named option1CheckBox. The following if statement determines whether it is selected:

```
if (option1CheckBox.Checked)
{
   MessageBox.Show("You selected Option 1.");
}
```

The application's form is shown in Figure 4-5. When you run the application, select any of the check boxes and then click the OK button. One or more message boxes will appear, showing you the items that you selected.

Figure 4-5 The CheckBox Example form



Here is the code for the okButton_Click event handler:

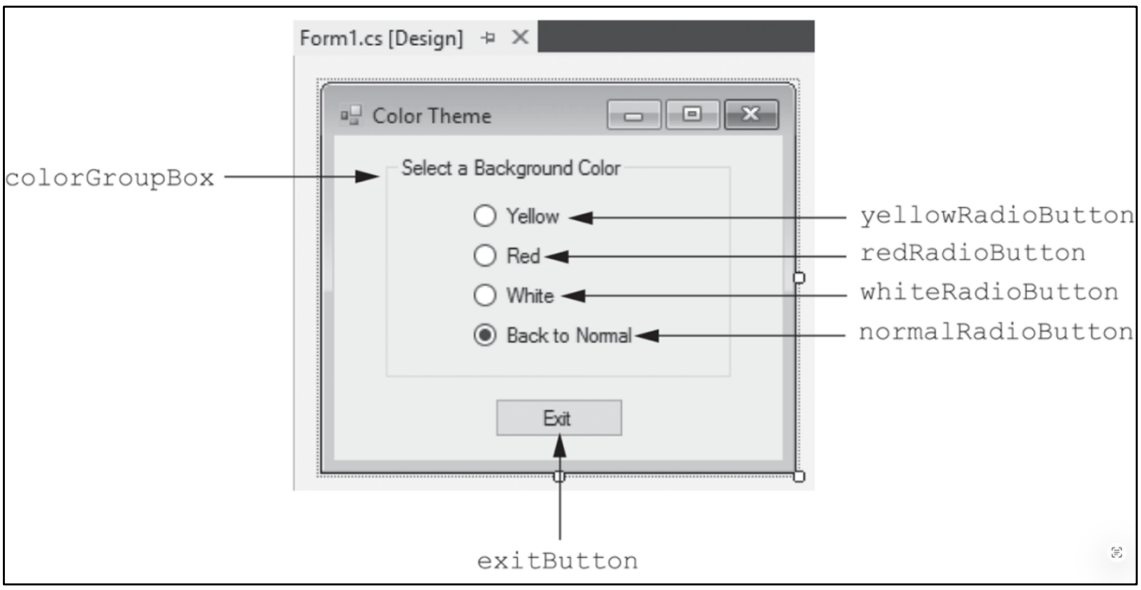
```
1        private void okButton_Click(object sender, EventArgs e)
2        {
3            if (pepperoniCheckBox.Checked)
4            {
```

```
5                       MessageBox.Show("You selected Pepperoni.");
6                   }
7
8                   if (cheeseCheckBox.Checked)
9                   {
10                      MessageBox.Show("You selected Cheese.");
11                  }
12
13                  if (anchoviesCheckBox.Checked)
14                  {
15                      MessageBox.Show("You selected Anchovies.");
16                  }
17              }
```

Creating the Color Theme Application

Create an application that allows the user to select a color using RadioButton controls. When the user selects a color, the form's background color is changed to that color immediately. As Shown in Figure 4-6.

Figure 4-6 The Color Theme form




```
11 namespace Color_Theme
12                     {
13                     public partial class Form1 : Form
14                         {
15                         public Form1()
16                             {
17                             InitializeComponent();
18                         }
19
20     private void yellowRadioButton_CheckedChanged(object sender, EventArgs e)
21                     {
```

```
22                          if (yellowRadioButton.Checked)
23                          {
24                                  this.BackColor = Color.Yellow;
25                          }
26                  }
27
28      private void redRadioButton_CheckedChanged(object sender, EventArgs e)
29      {
30                          if (redRadioButton.Checked)
31                          {
32                                  this.BackColor = Color.Red;
33                          }
34      }
35
36      private void whiteRadioButton_CheckedChanged(object sender, EventArgs e)
37      {
38                  if (whiteRadioButton.Checked)
39                  {
40                          this.BackColor = Color.White;
41                  }
```

```
42      }
43
44      private void normalRadioButton_CheckedChanged(object sender, EventArgs e)
45      {
46                      if (normalRadioButton.Checked)
47                      {
48                              this.BackColor = SystemColors.Control;
49                      }
50      }
51
52      private void exitButton_Click(object sender, EventArgs e)
53      {
54          // Close the form.
55          this.Close();
56      }
57  }
58 }
```

Exercise: Develop a C# Windows Forms application that allows the user to select specific mathematical operations using CheckBox controls to perform calculations on two numbers.