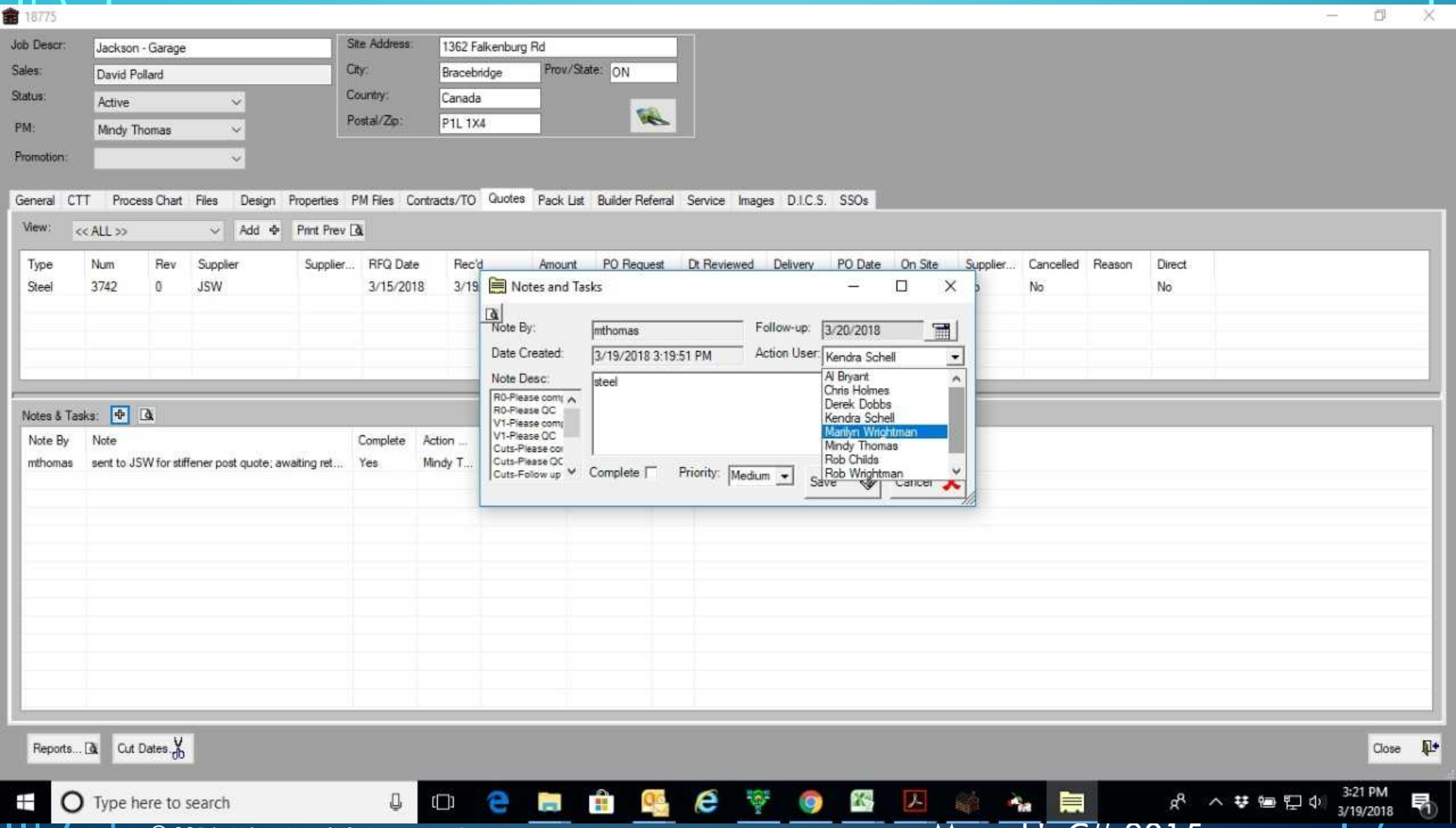
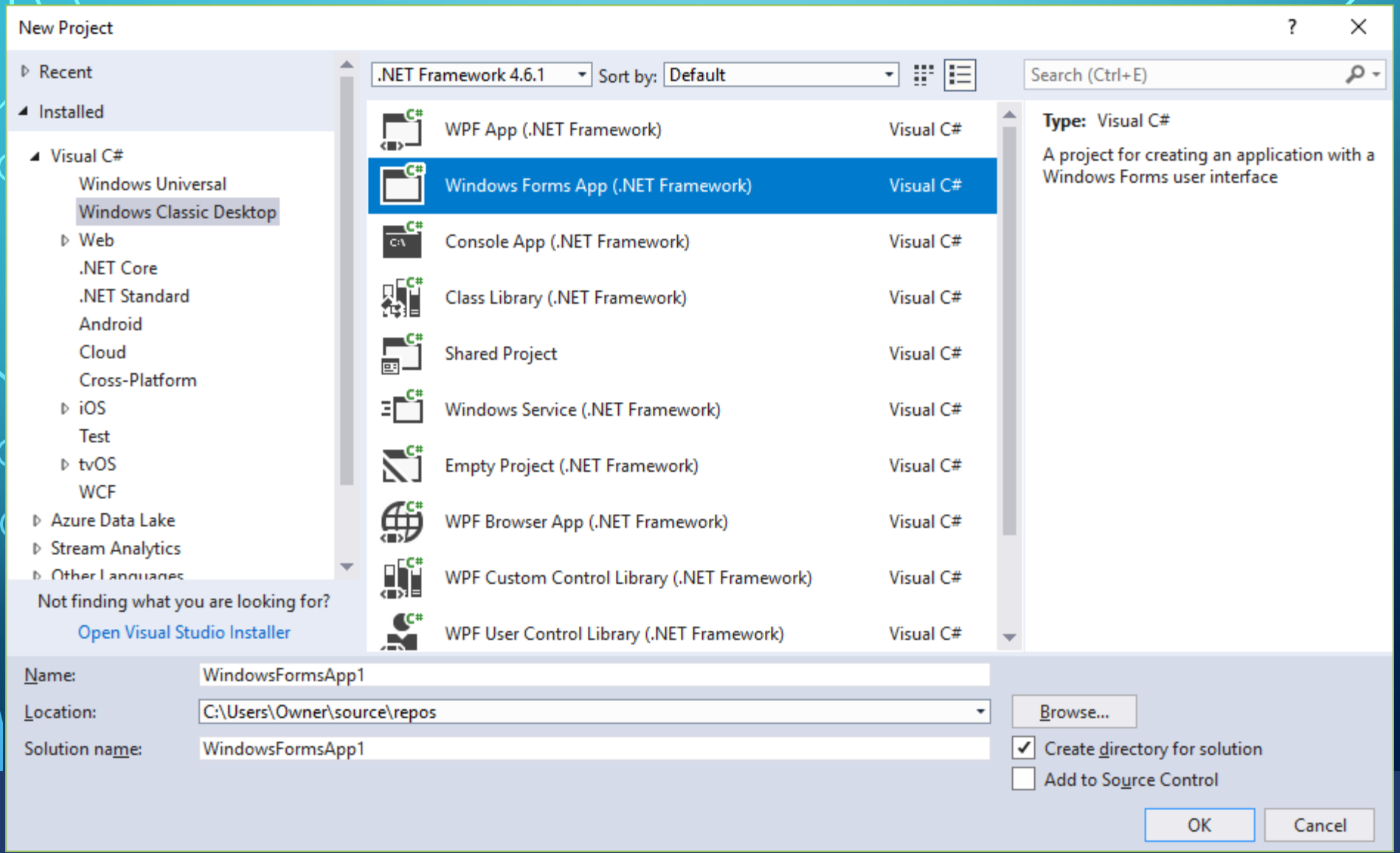
CSHARP

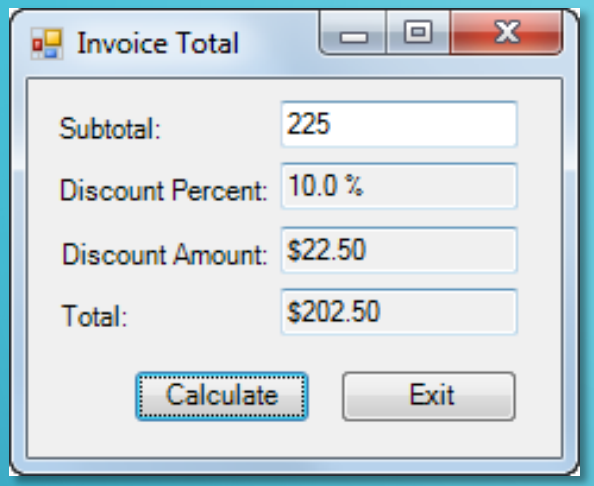
Windows forms and controls


18775
Job Descr: Jackson - Garage
Sales: David Pollard
Status: Active
PM: Mindy Thomas
Promotion:
Site Address: 1362 Falkenburg Rd
City: Bracebridge
Prov/State: ON
Country: Canada
Postal/Zip: P1L 1X4
General CTT Process Chart Files Design Properties PM Files Contracts/TO Quotes Pack List Builder Referral Service Images D.I.C.S. SSOs
View: << ALL >>
Add
Print Prev
Type Num Rev Supplier Supplier... RFQ Date Rec'd Amount PO Request Dt Reviewed Delivery PO Date On Site Supplier... Cancelled Reason Direct
Steel 3742 0 JSW 3/15/2018 No No
Notes and Tasks
Note By: mthomas
Follow-up: 3/20/2018
Date Created: 3/19/2018 3:19:51 PM
Action User: Kendra Schell
Note Desc:
steel
Al Bryant
Chris Holmes
Derek Dobbs
Kendra Schell
Marilyn Wrightman
Mindy Thomas
Rob Childs
Rob Wrightman
Complete
Priority: Medium
Save
Cancel
Notes & Tasks:
Note By Note Complete Action ...
mthomas sent to JSW for stiffener post quote; awaiting ret... Yes Mindy T...
Reports...
Cut Dates
Close
Type here to search
3:21 PM
3/19/2018

THE NEW PROJECT DIALOG BOX (VS WIN FORM CLASS)



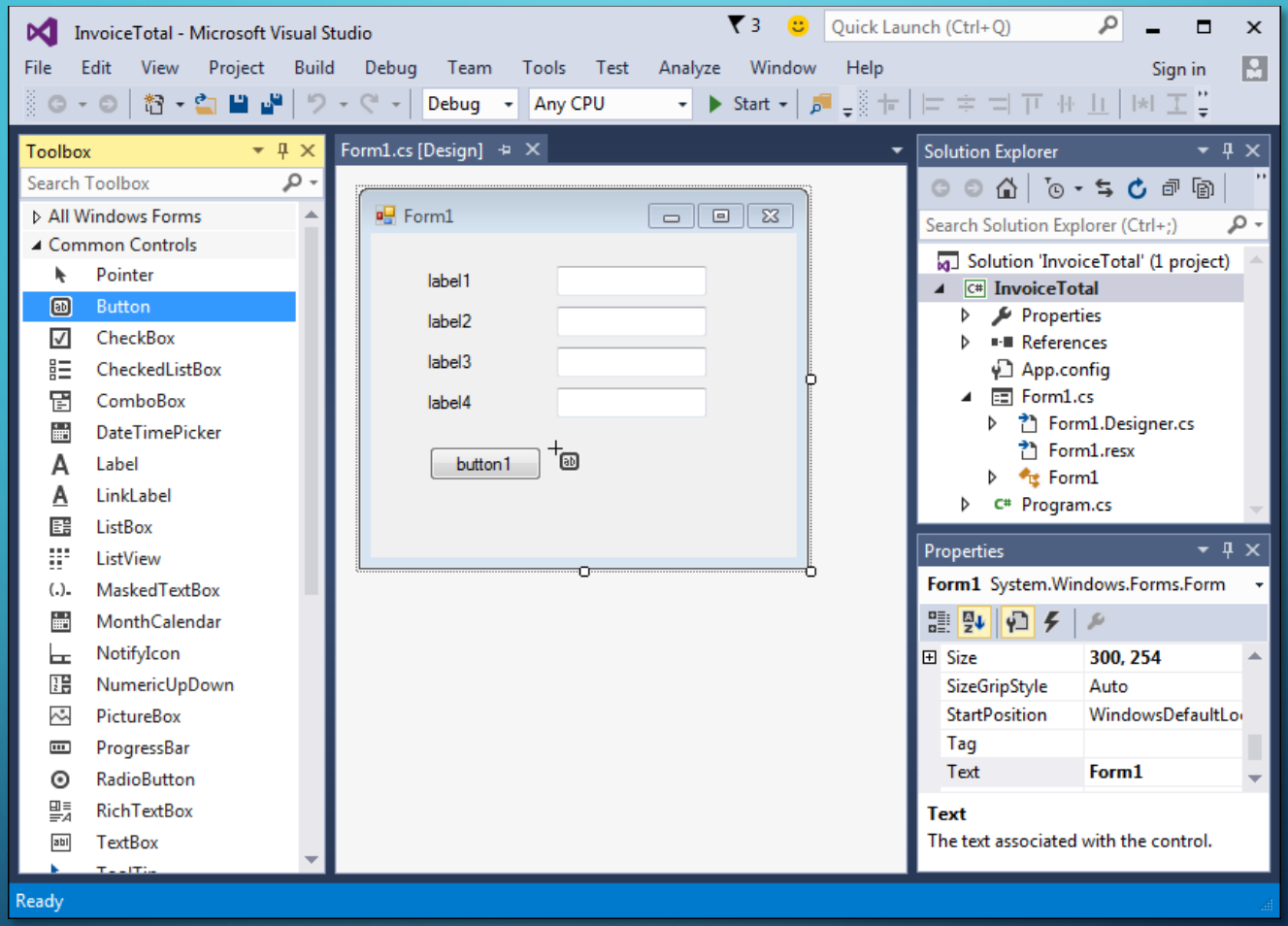
THE INVOICE TOTAL FORM



Three types of controls

- A *label* displays text on a form.
- A *text box* lets the user enter text on a form.
- A *button* initiates form processing when clicked.

A FORM AFTER SOME CONTROLS HAVE BEEN ADDED TO IT



THREE WAYS TO ADD A CONTROL TO A FORM

- Select the control in the Toolbox. Then, click in the form where you want to place the control. Or, drag the pointer on the form to place the control and size it at the same time.
- Double-click the control in the Toolbox. Then, the control is placed in the upper left corner of the form.
- Drag the control from the Toolbox and drop it on the form. Then, the control is placed wherever you drop it.

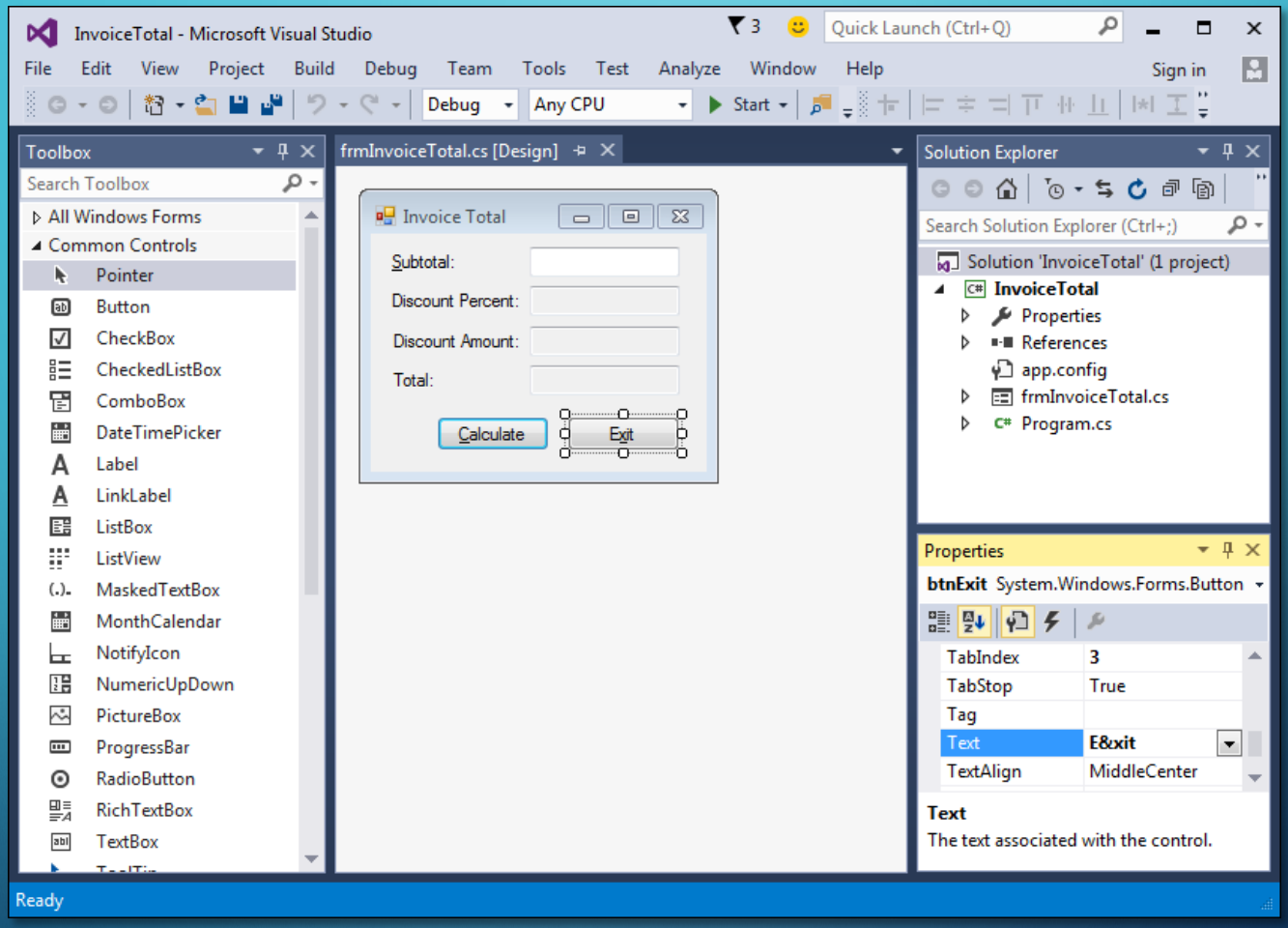
HOW TO SELECT AND WORK WITH CONTROLS

- Use standard Windows techniques to select, move, size, or group the controls on a form.
- You can also select a group of controls by clicking on a blank spot in the form and then dragging around the controls.
- To align, size, or space a group of selected controls, click on a control to make it the *primary control*. Then, use the commands in the Format menu or the buttons on the Layout toolbar to align, size, or space the controls relative to the primary control.
- You can also size all of the controls in a group by sizing the primary control in the group.

Note

- A label is sized automatically based on the amount of text that it contains. As a result, you can't size a label by dragging its handles unless you change its AutoSize property to False.

A FORM AFTER THE PROPERTIES HAVE BEEN SET



THE NAME PROPERTY

- Sets the name you use to identify a control in your C# code.
- Can be changed to provide a more descriptive and memorable name for forms and controls that you will refer to when you write your code (such as text boxes and buttons).
- Doesn't need to be changed for controls that you won't refer to when you write your code (such as most labels).
- Can use a three-letter prefix to indicate whether the name refers to a form (frm), button (btn), label (lbl), or text box (txt).

THE TEXT PROPERTY

- Sets the text that's displayed on the form or control.
- For a form, the Text value is displayed in the title bar. For controls, the Text value is displayed directly on the control.
- For a text box, the Text value changes when the user types text into the control, and you can write code that uses the Text property to get the text that was entered by the user.

OTHER PROPERTIES FOR FORMS

Property	Description
`AcceptButton`	Identifies the button that will be activated when the user presses the Enter key.
`CancelButton`	Identifies the button that will be activated when the user presses the Esc key.
`StartPosition`	Sets the position at which the form is displayed. To center the form, set this property to CenterScreen.

OTHER PROPERTIES FOR CONTROLS

Property	Description
`Enabled`	Determines whether the control will be enabled or disabled.
`ReadOnly`	Determines whether the text in some controls like text boxes can be edited.
`TabIndex`	Indicates the control's position in the tab order, which determines the order in which the controls will receive the focus when the user presses the Tab key.
`TabStop`	Determines whether the control will accept the focus when the user presses the Tab key to move from one control to another. Some controls, like labels, don't have the TabStop property.
`TextAlign`	Sets the alignment for the text displayed on a control.

HOW TO ADJUST THE TAB ORDER

- *Tab order* refers to the sequence in which the controls receive the *focus* when the user presses the Tab key. You should adjust the tab order so the Tab key moves the focus from one control to the next in a logical sequence.
- Each control has a TabIndex property that indicates the control's position in the tab order. You can change this property to change a control's tab order position.
- If you don't want a control to receive the focus when the user presses the Tab key, change that control's TabStop property to False.
- Label controls don't have a TabStop property so they can't receive the focus.

HOW TO SET ACCESS KEYS

- *Access keys* are shortcut keys that the user can use in combination with the Alt key to quickly move to individual controls on the form.
- You use the Text property to set the access key for a control by placing an ampersand immediately before the letter you want to use for the access key.
- Since the access keys aren't case sensitive, &N and &n set the same access key.
- When you set access keys, make sure to use a unique letter for each control.
- You can't set the access key for a text box. However, if you set an access key for a label that immediately precedes the text box in the tab order, the access key will take the user to the text box.
- If you assign an access key to a button, the button is activated when you press Alt plus the access key.

HOW TO SET THE ENTER AND ESC KEYS

- The AcceptButton property of the form sets the button that will be activated if the user presses the Enter key.
- The CancelButton property of the form sets the button that will be activated if the user presses the Esc key. This property should usually be set to the Exit button.
- You set the AcceptButton or CancelButton values by choosing the button from a drop-down list that shows all of the buttons on the form. So be sure to create and name the buttons you want to use before you attempt to set these values.

THE PROPERTY SETTINGS FOR THE INVOICE TOTAL FORM

Default name	Property	Setting
Form1	Text	Invoice Total
	AcceptButton	btnCalculate
	CancelButton	btnExit
	StartPosition	CenterScreen

THE PROPERTY SETTINGS FOR THE CONTROLS

Default name	Property	Setting
label1	Text	&Subtotal:
	TextAlign	MiddleLeft
	TabIndex	0
label2	Text	Discount percent:
	TextAlign	MiddleLeft
label3	Text	Discount amount:
	TextAlign	MiddleLeft
label4	Text	Total:
	TextAlign	MiddleLeft
textBox1	Name	txtSubtotal
	TabIndex	1
textBox2	Name	txtDiscountPercent
	ReadOnly	True
	TabStop	False

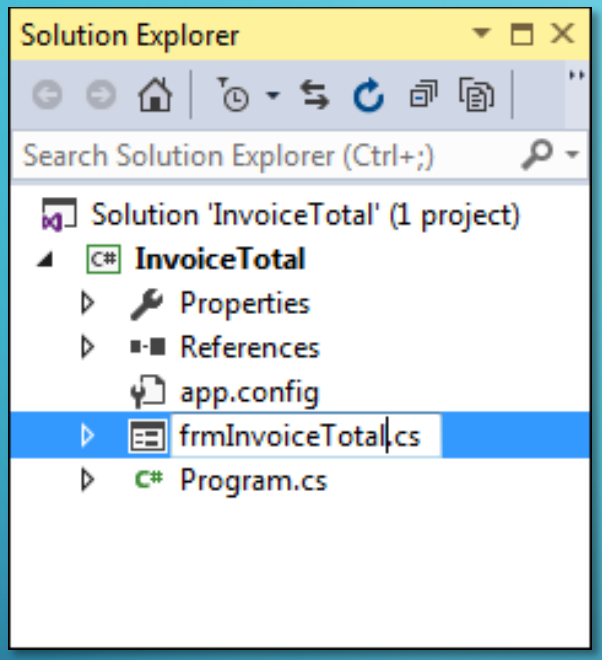
THE PROPERTY SETTINGS FOR THE CONTROLS (CONT.)

textBox3	Name	txtDiscountAmount
	ReadOnly	True
	TabStop	False
textBox4	Name	txtTotal
	ReadOnly	True
	TabStop	False
button1	Name	btnCalculate
	Text	&Calculate
	TabIndex	2
button2	Name	btnExit
	Text	E&xit
	TabIndex	3

How to rename a file, project, or solution (if you copied the exact codes of assignment samples)

- Right-click on it in the Solution Explorer window and select the Rename command from the shortcut menu. Or, you can select it in the Solution Explorer and press F2. Then, you can enter the new name.
- Be sure not to change or omit the file extension when you rename a file. Remember too that using a three-letter prefix to indicate the contents of the file (like *frm* for a form file) makes it easier to tell what each file represents.
- When you change the name of a file, Visual Studio also changes the File Name property for the form and asks you if you want to change all references to the file.
- If you change the name of a project, you may also want to change the name of the assembly that's created for the project and the name of the namespace that contains the project from the Application tab of the Project Properties window.

THE SOLUTION EXPLORER AS A FORM FILE IS BEING RENAMED



HOW TO SAVE A FILE, PROJECT, OR SOLUTION

- You can use the Save All button in the Standard toolbar or the Save All command in the File menu to save all files and projects in the solution.
- You can use the Save button in the Standard toolbar or the Save command in the File menu to save a file, project, or solution. The files that are saved depend on what's selected in the Solution Explorer window.
- If you try to close a solution that contains modified files, a dialog box is displayed that asks you if you want to save those files.

OBJECTS AND FORMS

- When you use the Form Designer, Visual Studio automatically generates C# code that creates a new class based on the Form class. Then, when you run the project, a form object is instantiated from the new class.
- When you add a control to a form, Visual Studio automatically generates C# code in the class for the form that instantiates a control object from the appropriate class and sets the control's default properties.
- When you move and size a control, Visual Studio automatically sets the properties that specify the location and size of the control.

THE SYNTAX FOR REFERRING TO A MEMBER OF A CLASS OR OBJECT

```
ClassName.MemberName
objectName.MemberName
```

Statements that refer to properties

`txtTotal.Text = "10";`	Assigns a string holding the number 10 to the Text property of the text box named txtTotal.
`txtTotal.ReadOnly = true;`	Assigns the true value to the ReadOnly property of the text box named txtTotal so the user can't change its contents.

STATEMENTS THAT REFER TO METHODS

`txtMonthlyInvestment.Focus();`	Uses the Focus method to move the focus to the text box named txtMonthlyInvestment.
`this.Close();`	Uses the Close method to close the form that contains the statement. Here, *this* is a keyword that is used to refer to the current instance of the class.

Code that refers to an event

`btnExit.Click`	Refers to the Click event of a button named btnExit.

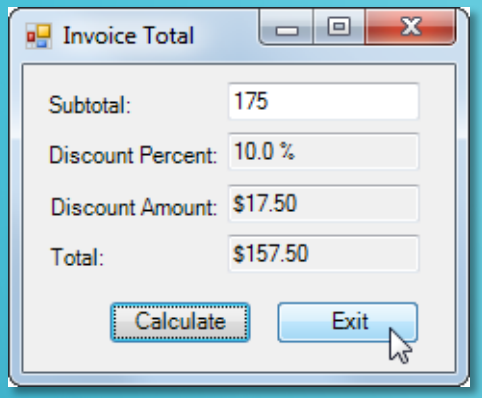
HOW TO ENTER MEMBER NAMES WHEN WORKING IN THE CODE EDITOR

1. To display a list of the available members for a class or an object, type the class or object name followed by a period (called a *dot operator*, or just *dot*).
2. Type one or more letters of the member name, and the Code Editor will select the first entry in the list that matches those letters. Or, scroll down the list to select the member you want.
3. Press the Tab or Enter key to insert the member into your code.

Note

- If a member list isn't displayed, select the Tools→Options command to display the Options dialog box. Then, expand the Text Editor group, select the C# category, and check the Auto List Members and Parameters Information boxes.

EVENT: THE USER CLICKS THE EXIT BUTTON



Wiring: The application determines what method to execute

```
this.btnExit.Click +=
    new System.EventHandler(this.btnExit_Click);
```

Response: The method for the Click event of the Exit button is executed

```
private void btnExit_Click(object sender, EventArgs e)
{
    this.Close();
}
```

COMMON CONTROL EVENTS

Event	Occurs when…
`Click`	…the user clicks the control.
`DoubleClick`	…the user double-clicks the control.
`Enter`	…the focus is moved to the control.
`Leave`	…the focus is moved from the control.

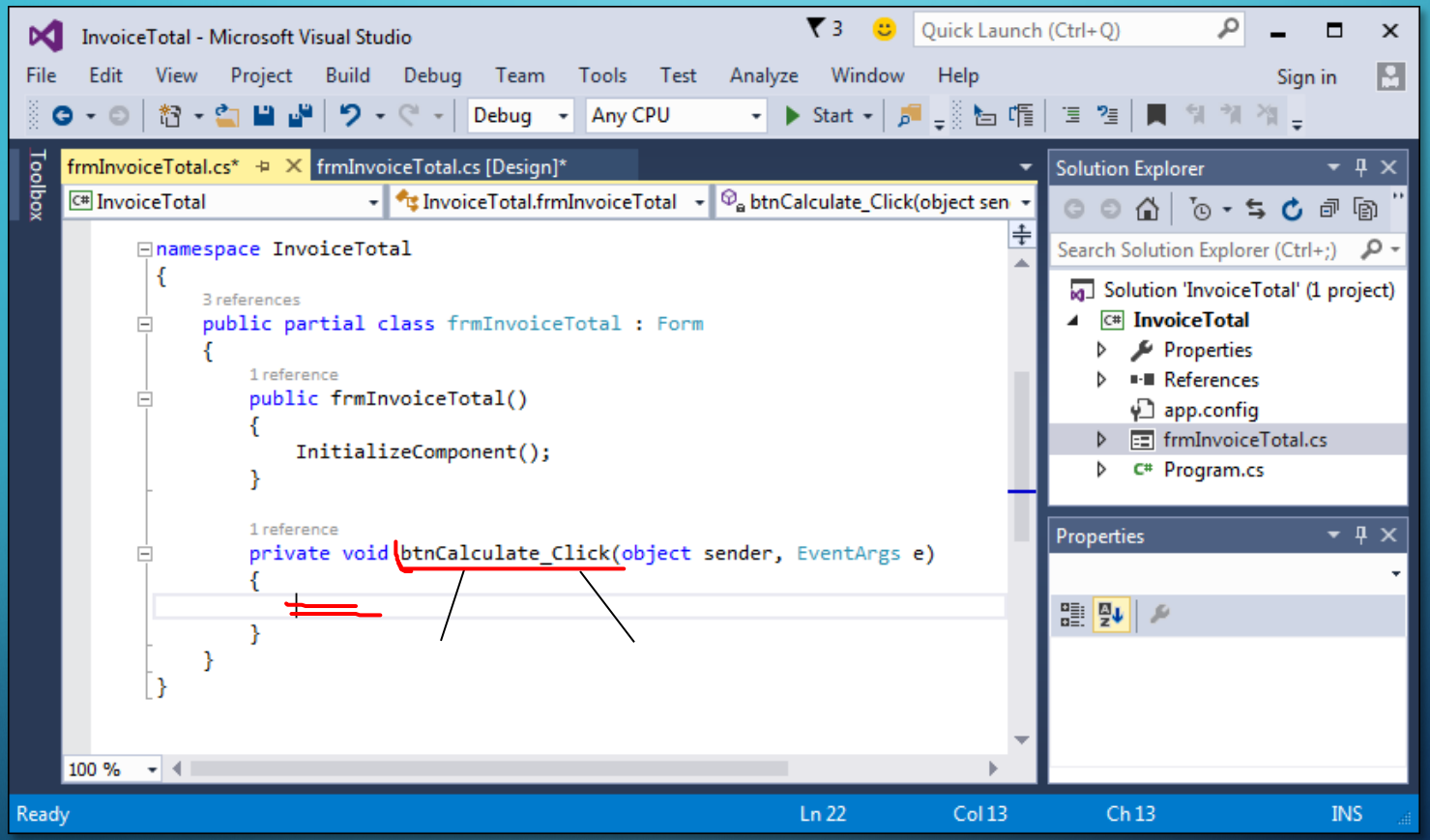
Common form events

Event	Occurs when…
`Load`	…the form is loaded into memory.
`Closing`	…the form is closing.
`Closed`	…the form is closed.

HOW AN APPLICATION RESPONDS TO EVENTS

- Windows Forms applications work by responding to events that occur on objects.
- To indicate how an application should respond to an event, you code an *event handler*, which is a special type of method that handles the event.
- To connect the event handler to the event, Visual Studio automatically generates a statement that wires the event to the event handler. This is known as *event wiring*.
- An event can be an action that's initiated by the user like the Click event, or it can be an action initiated by program code like the Closed event.

THE METHOD THAT HANDLES THE CLICK EVENT OF THE CALCULATE BUTTON



HOW TO HANDLE THE CLICK EVENT OF A BUTTON

1. In the Form Designer, double-click the control. This opens the Code Editor, generates the declaration for the method that handles the event, and places the cursor within this declaration.
2. Type the C# code between the opening brace ({) and the closing brace (}) of the method declaration.
3. When you are finished writing code, you can return to the Form Designer by clicking on its tab.

How to handle the Load event for a form

- Follow the procedure above, but double-click the form itself.

THE EVENT HANDLERS FOR THE INVOICE TOTAL FORM

```
private void btnCalculate_Click(object sender,
EventArgs e)
{
    decimal subtotal =
        Convert.ToDecimal(txtSubtotal.Text);
    decimal discountPercent = 0m;
    if (subtotal >= 500)
    {
        discountPercent = .2m;
    }
    else if (subtotal >= 250 && subtotal < 500)
    {
        discountPercent = .15m;
    }
    else if (subtotal >= 100 && subtotal < 250)
    {
        discountPercent = .1m;
    }

    decimal discountAmount = subtotal * discountPercent;
    decimal invoiceTotal = subtotal - discountAmount;
```

THE EVENT HANDLERS FOR THE INVOICE TOTAL FORM (CONT.)

```
        txtDiscountPercent.Text =
            discountPercent.ToString("p1");
        txtDiscountAmount.Text =
            discountAmount.ToString("c");
        txtTotal.Text = invoiceTotal.ToString("c");

        txtSubtotal.Focus();
    }

    private void btnExit_Click(object sender, EventArgs e)
    {
        this.Close();
    }
```

A METHOD WRITTEN IN A READABLE STYLE

```
private void btnCalculate_Click(object sender,
EventArgs e)
{
    decimal subtotal =
        Convert.ToDecimal(txtSubtotal.Text);

    decimal discountPercent = 0m;
    if (subtotal >= 500)
    {
        discountPercent = .2m;
    }
    else if (subtotal >= 250 && subtotal < 500)
    {
        discountPercent = .15m;
    }
    else if (subtotal >= 100 && subtotal < 250)
    {
        discountPercent = .1m;
    }
```

A METHOD WRITTEN IN A READABLE STYLE (CONT.)

```
        decimal discountAmount = subtotal * discountPercent;
        decimal invoiceTotal = subtotal - discountAmount;

        txtDiscountPercent.Text =
            discountPercent.ToString("p1");
        txtDiscountAmount.Text =
            discountAmount.ToString("c");
        txtTotal.Text = invoiceTotal.ToString("c");

        txtSubtotal.Focus();
    }
```

A METHOD WRITTEN IN A LESS READABLE STYLE

```
private void btnCalculate_Click(object sender, EventArgs e){
decimal subtotal=Convert.ToDecimal(txtSubtotal.Text);
decimal discountPercent=0m;
if (subtotal>=500) discountPercent=.2m;
else if (subtotal>=250&&subtotal<500) discountPercent=.15m;
else if (subtotal>=100&&subtotal<250) discountPercent=.1m;
decimal discountAmount=subtotal*discountPercent;
decimal invoiceTotal=subtotal-discountAmount;
txtDiscountPercent.Text=discountPercent.ToString("p1");
txtDiscountAmount.Text=discountAmount.ToString("c");
txtTotal.Text=invoiceTotal.ToString("c");txtSubtotal.Focus();}
```

CODING RECOMMENDATIONS

- Use indentation and extra spaces to align statements and blocks of code so they reflect the structure of the program.
- Use spaces to separate the words, operators, and values in each statement.
- Use blank lines before and after groups of related statements.

Note

- As you enter code in the Code Editor, Visual Studio automatically adjusts its formatting by default.

A METHOD WITH COMMENTS

```
private void btnCalculate_Click(object sender, EventArgs e)
{
    /*****************************************
    * this method calculates the total
    * for an invoice depending on a
    * discount that's based on the subtotal
    *****************************************/

    // get the subtotal amount from the Subtotal text box
    decimal subtotal = Convert.ToDecimal(txtSubtotal.Text);

    // set the discountPercent variable based
    // on the value of the subtotal variable
    decimal discountPercent = 0m;        // the m indicates
                                         // a decimal value
    if (subtotal >= 500)
    {
        discountPercent = .2m;
    }
    else if (subtotal >= 250 && subtotal < 500)
    {
        discountPercent = .15m;
    }
```

OLD STYLE (Bad)

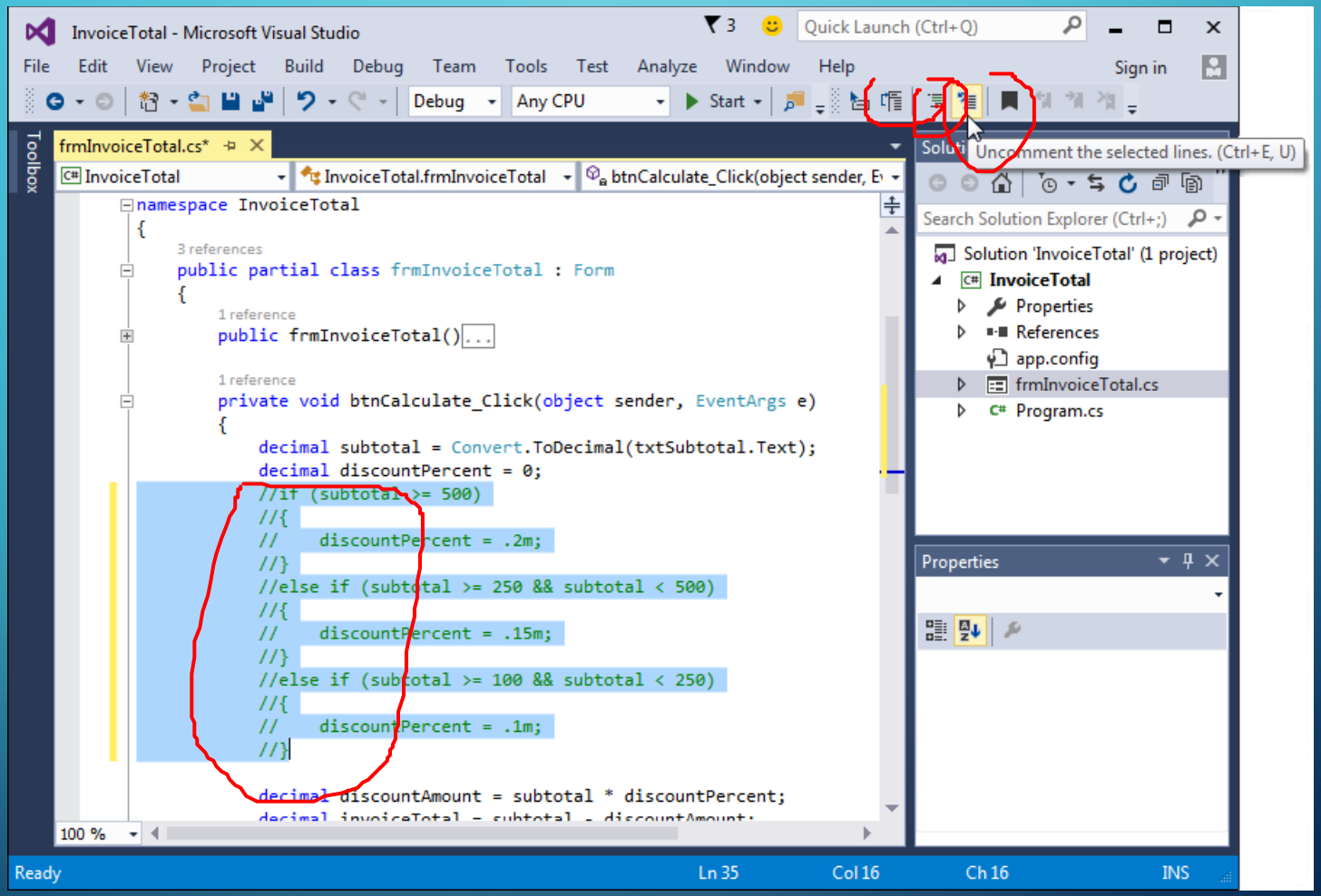
A METHOD WITH COMMENTS (CONT.)

```
else if (subtotal >= 100 && subtotal < 250)
    {
        discountPercent = .1m;
    }

    // calculate and assign the values for the
    // discountAmount and invoiceTotal variables
    decimal discountAmount = subtotal * discountPercent;
    decimal invoiceTotal = subtotal - discountAmount;

    // format the values and display them in their text boxes
    txtDiscountPercent.Text =              // percent format
        discountPercent.ToString("p1");  // with 1 decimal place
    txtDiscountAmount.Text =
        discountAmount.ToString("c");    // currency format
    txtTotal.Text =
        invoiceTotal.ToString("c");

    // move the focus to the Subtotal text box
    txtSubtotal.Focus();
}
```

HOW TO CODE COMMENTS

- *Comments* are used to help document what a program does and what the code within it does.
- To code a *single-line comment*, type // before the comment. You can use this technique to add a comment on its own line or to add a comment at the end of a line.
- To code a *delimited comment*, type /* at the start of the comment and */ at the end. You can also code asterisks to identify the lines in the comment, but that isn't necessary.

THE CODE EDITOR AND THE TEXT EDITOR TOOLBAR



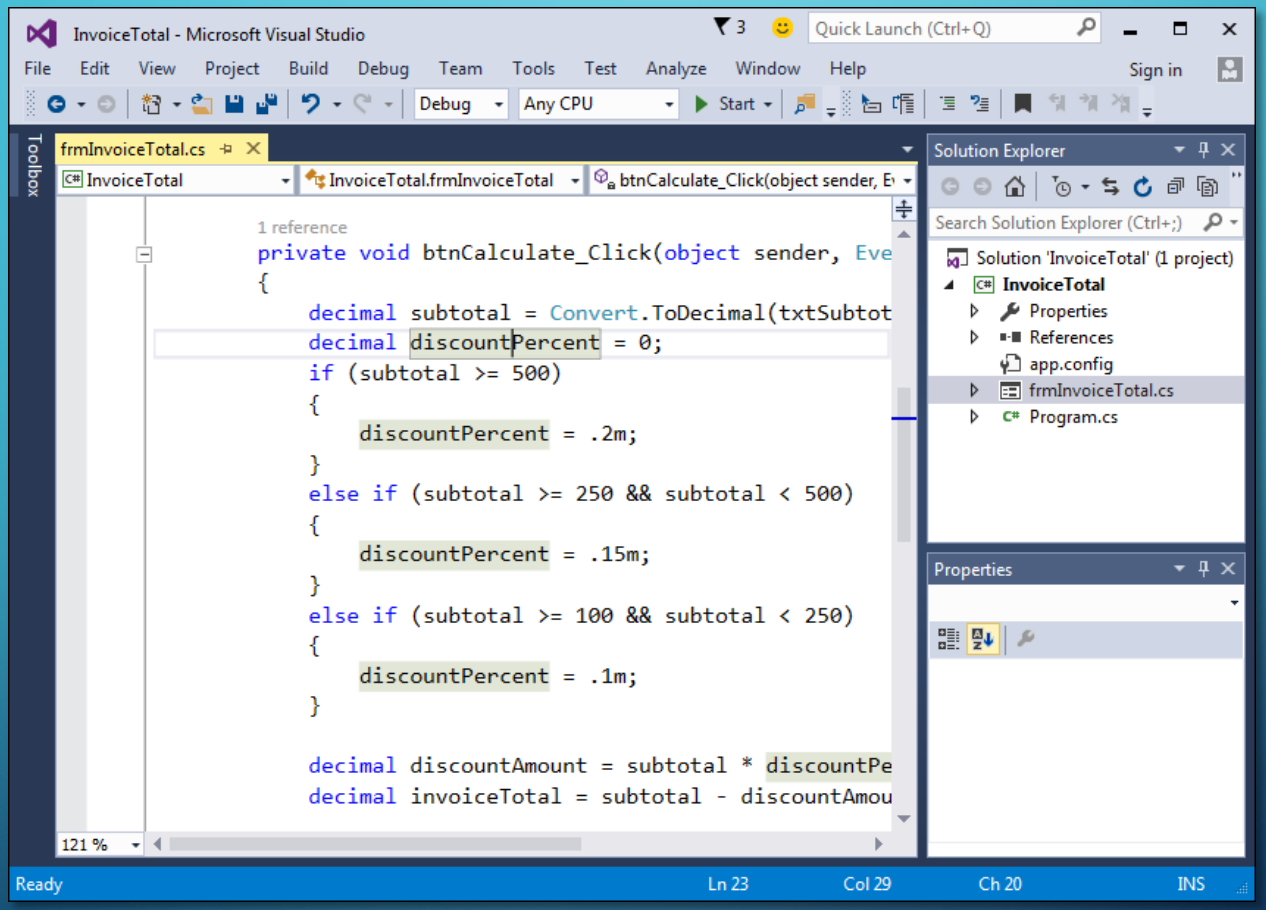
HOW TO USE THE BUTTONS OF THE TEXT EDITOR TOOLBAR

- To display or hide the Text Editor toolbar, right-click in the toolbar area and choose Text Editor from the shortcut menu.
- To comment or uncomment several lines of code, select the lines and click the Comment Out or Uncomment button.
- During testing, you can *comment out* a line of code. That way, you can test new statements without deleting the old statements.
- To move quickly between lines of code, you can use the last four buttons on the Text Editor toolbar to set and move between bookmarks.

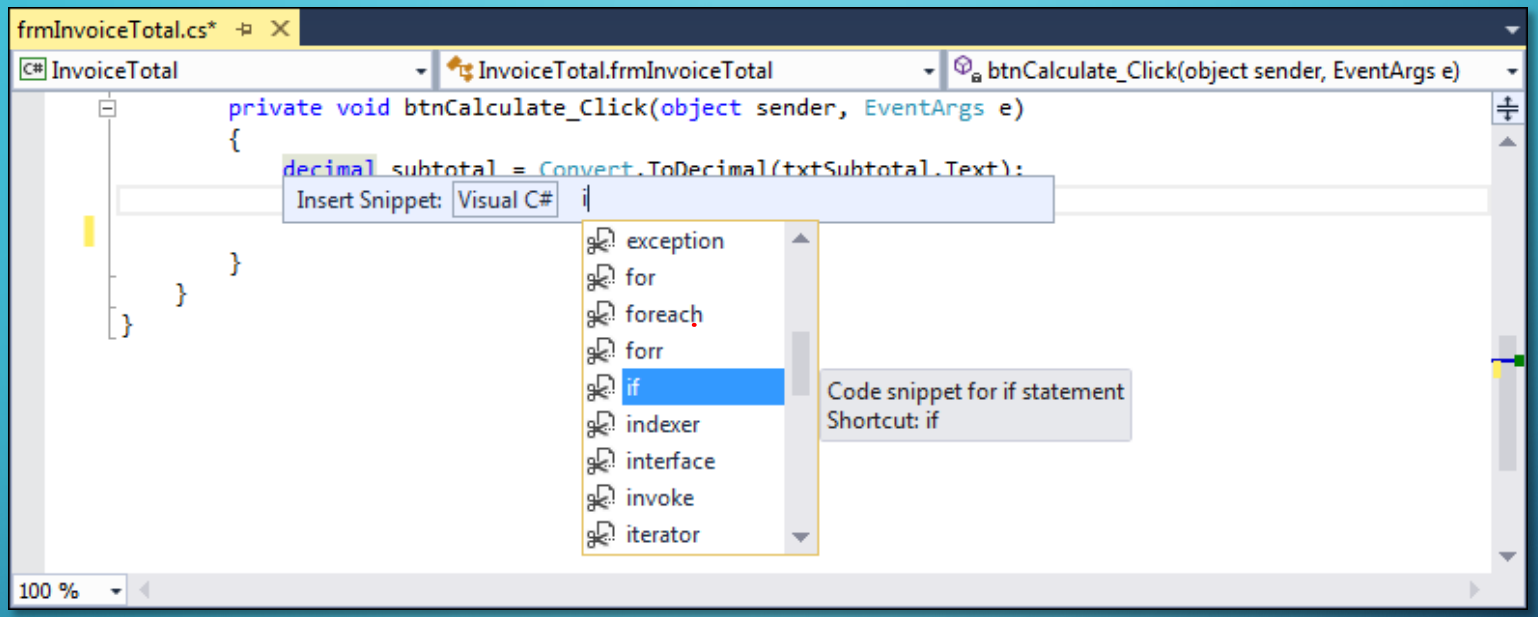
HOW TO COLLAPSE OR EXPAND REGIONS OF CODE

- If a region of code appears in the Code Editor with a minus sign (–) next to it, you can click the minus sign to collapse the region so just the first line is displayed.
- If a region of code appears in the Code Editor with a plus sign (+) next to it, you can click the plus sign to expand the region so all of its code is displayed.

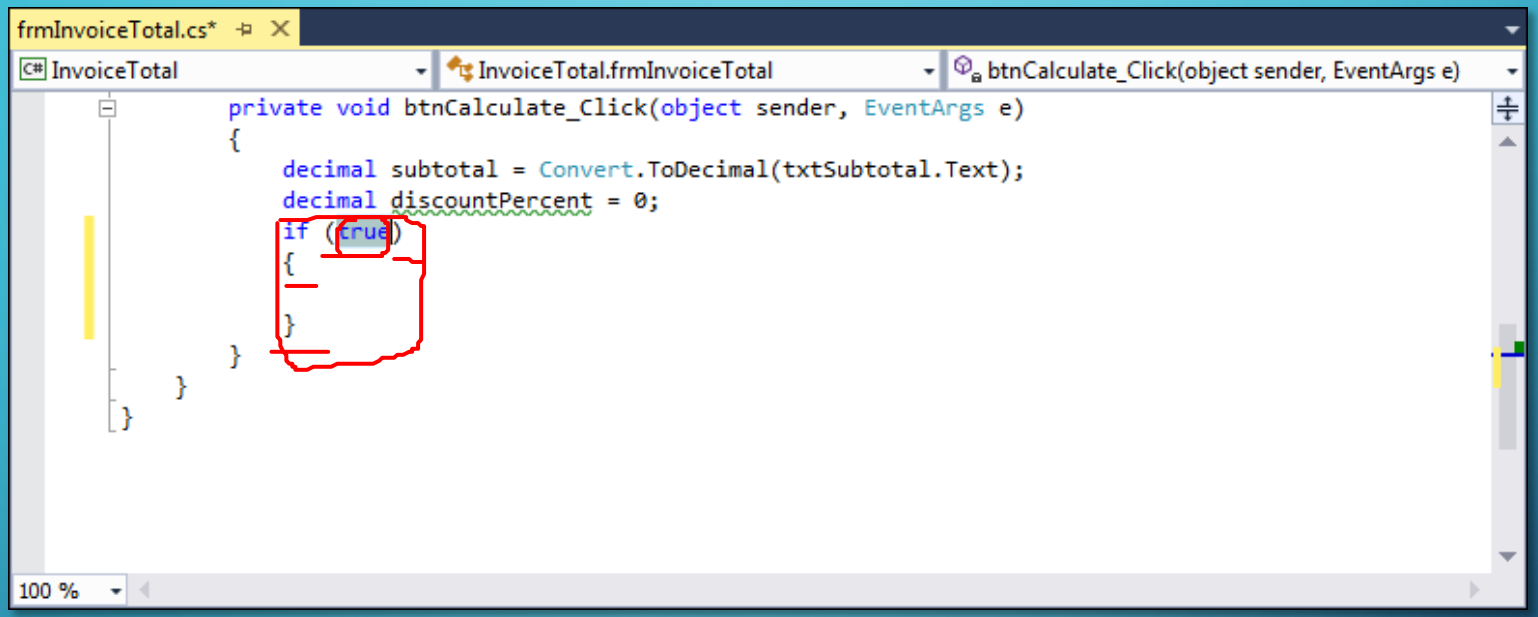
THE CODE EDITOR WITH ENLARGED TEXT AND A HIGHLIGHTED VARIABLE



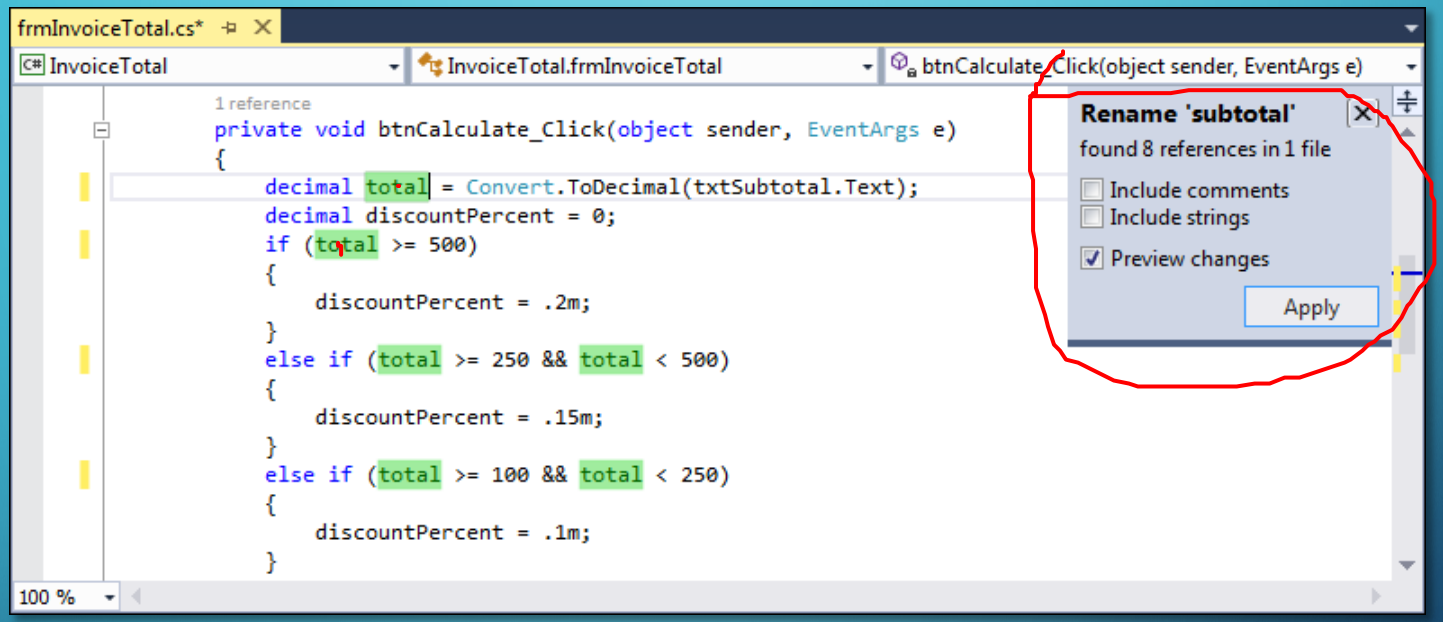
THE DEFAULT LIST OF VISUAL C# CODE SNIPPETS



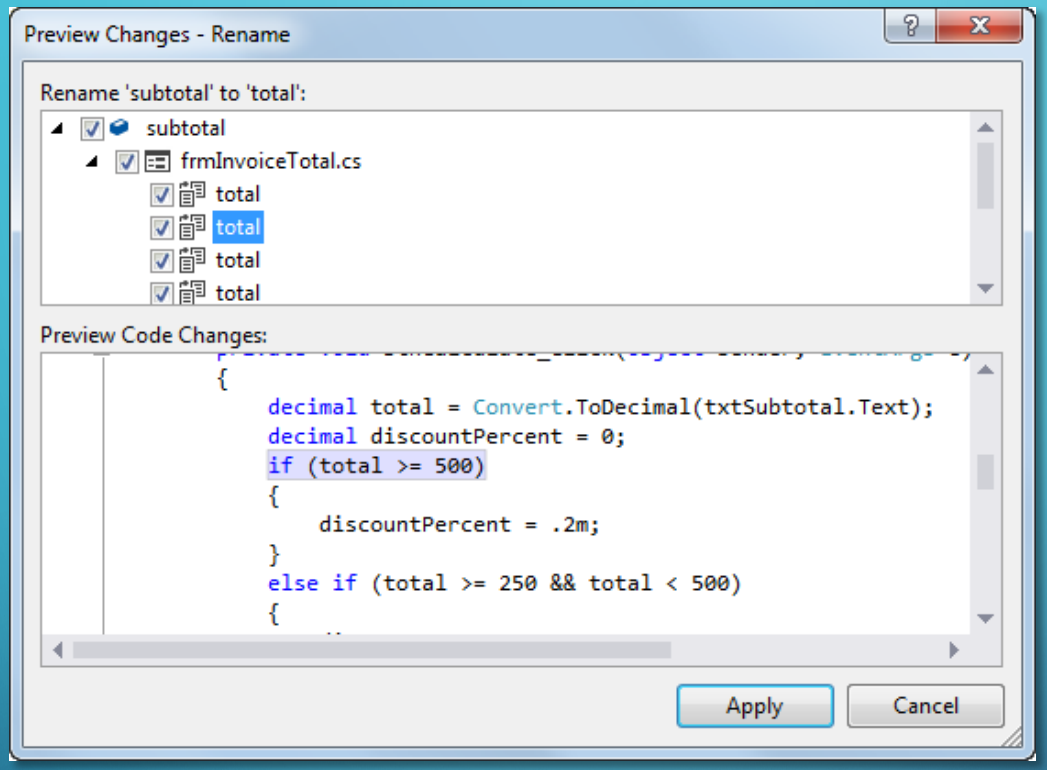
A CODE SNIPPET AFTER IT HAS BEEN INSERTED



THE OPTIONS THAT ARE DISPLAYED WHEN YOU RENAME A VARIABLE

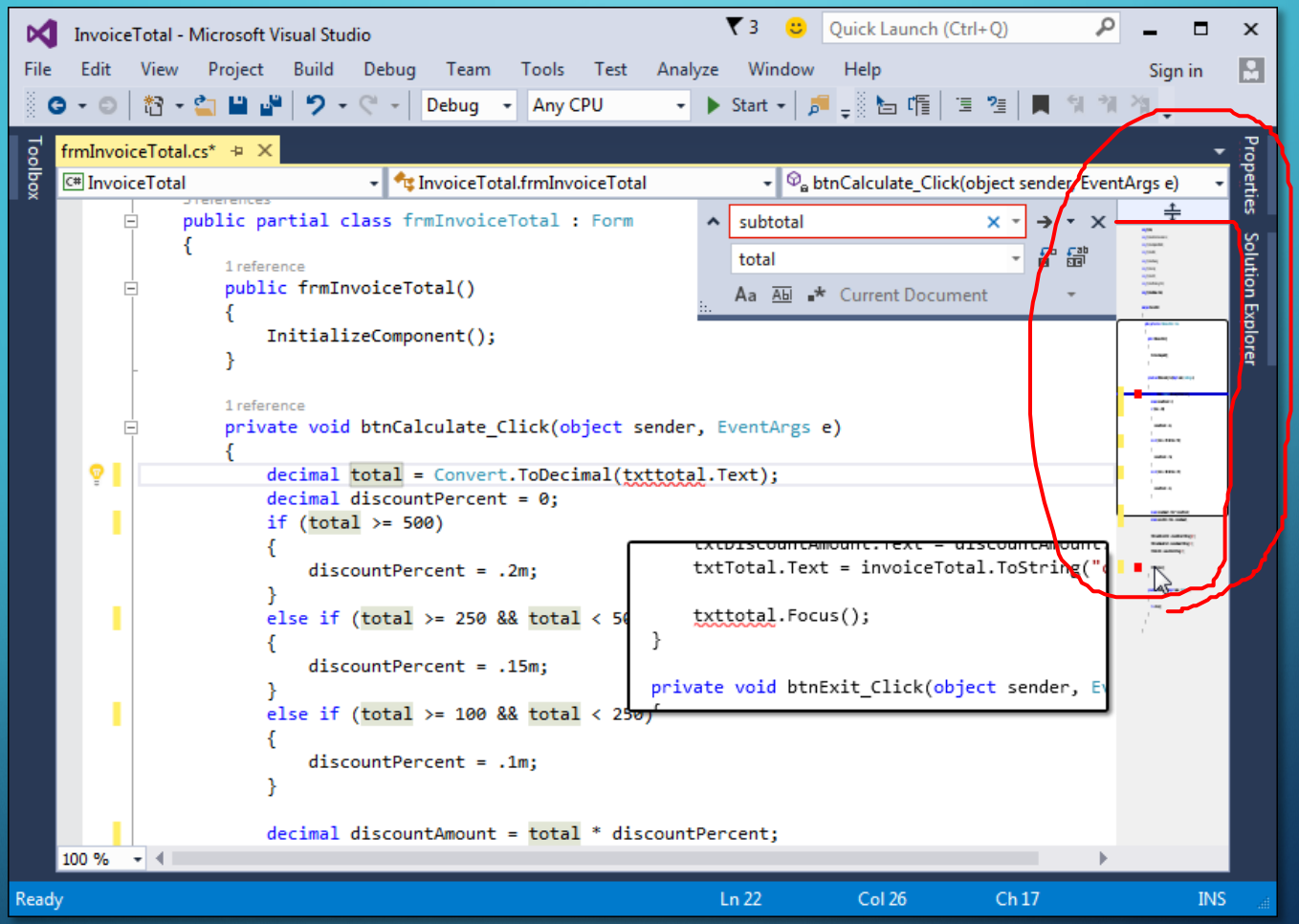


THE PREVIEW CHANGES - RENAME DIALOG BOX

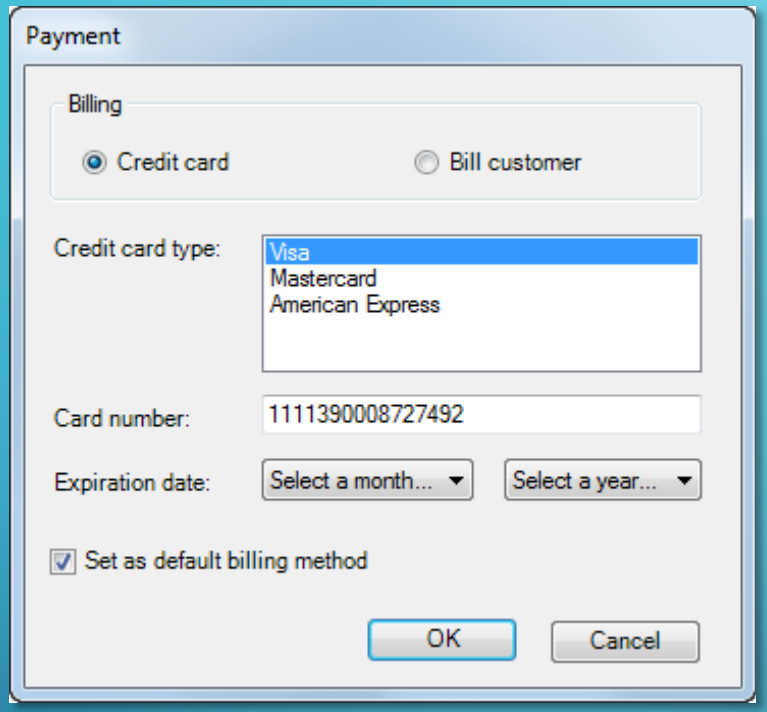


https://docs.microsoft.com/en-us/visualstudio/ide/how-to-track-your-code-by-customizing-the-scrollbar?view=vs-2017xt

A SCROLL BAR IN MAP MODE WITH ANNOTATIONS



A FORM WITH FIVE MORE TYPES OF CONTROLS



COMMON MEMBERS OF LIST BOX AND COMBO BOX CONTROLS

- `SelectedIndex`
- `SelectedItem`
- `Text`
- `Sorted`
- `Items`
- `DropDownStyle`
- `SelectionMode`

Common events of list box and combo box controls

- `SelectedIndexChanged`
- `TextChanged`

COMMON PROPERTIES OF THE ITEMS COLLECTION

```
[index]
Count
```

Common methods of the Items collection

```
Add(object)
Insert(index, object)
Remove(object)
RemoveAt(index)
Clear()
```

CODE THAT LOADS A COMBO BOX WITH MONTHS

```
string[] months =
    {"Select a month...",
    "January", "February", "March", "April",
    "May", "June", "July", "August",
    "September", "October", "November", "December"};

foreach (string month in months)
{
    cboExpirationMonth.Items.Add(month);
}
```

CODE THAT LOADS A COMBO BOX WITH YEARS

```
int year = DateTime.Today.Year;
int endYear = year + 8;
cboExpirationYear.Items.Add("Select a year...");
while (year < endYear)
{
    cboExpirationYear.Items.Add(year);
    year++;
}
```

CODE THAT CLEARS AND LOADS A LIST BOX OF CREDIT CARDS

```
lstCreditCardType.Items.Clear();
lstCreditCardType.Items.Add("Visa");
lstCreditCardType.Items.Add("Mastercard");
lstCreditCardType.Items.Add("American Express");
lstCreditCardType.SelectedIndex = 0;
```

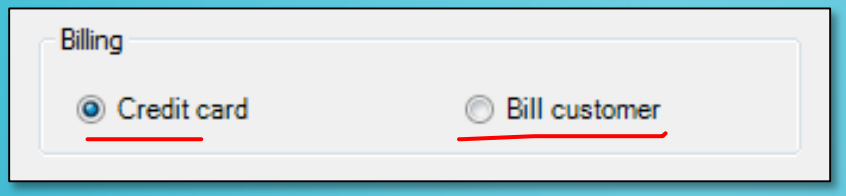
Statements that get information from a combo box or list box

```
int expYearIndex = cboExpirationYear.SelectedIndex;
string expYearText = cboExpirationYear.Text;
int expYearValue = (int) cboExpirationYear.SelectedItem;
string expMonthValue =
    cboExpirationMonth.Items[1].ToString();
```

CODE THAT WORKS WITH A COMBO BOX OF NAMES

```
string[] names = {"Judy Taylor", "Anne Boehm", "Kelly Slivkoff"};
foreach (string name in names)
{
    cboNames.Items.Add(name);
}
cboNames.Items.Insert(0, "Joel Murach");
cboNames.Items.RemoveAt(3);
cboNames.SelectedIndex = -1;       // don't select an item
```

A group box that contains two radio buttons



Common property of radio button and check box controls

`Checked`

Common event of radio button and check box controls

`CheckedChanged`

CODE THAT SETS THE VALUE OF A RADIO BUTTON AND CHECK BOX

```
rdoCreditCard.Checked = true;
chkDefault.Checked = true;
```

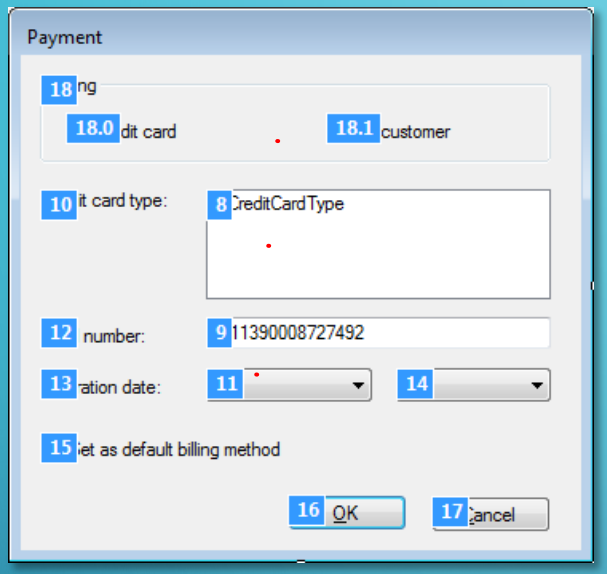
Code that checks the value of a radio button

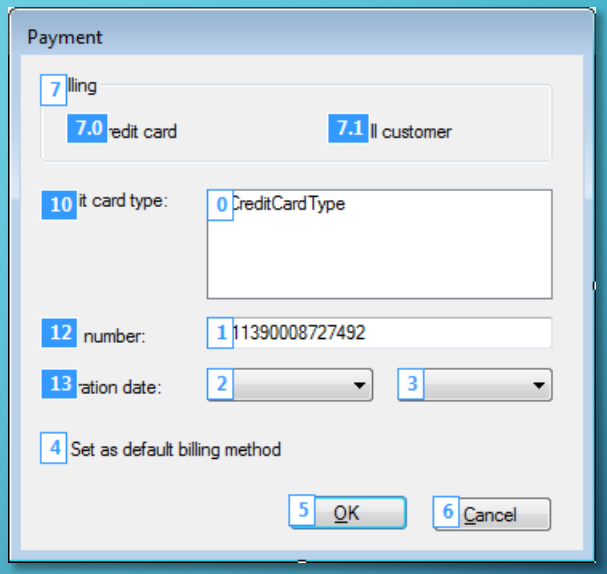
```
private void rdoCreditCard_CheckedChanged(object sender,
    EventArgs e)
{
    if (rdoCreditCard.Checked == true)
        EnableControls();
    else
        DisableControls();
}
```

Code that gets the value of a check box

```
bool isDefaultBilling = chkDefault.Checked;
```

A FORM IN TAB ORDER VIEW BEFORE AND AFTER THE TAB ORDER IS CHANGED

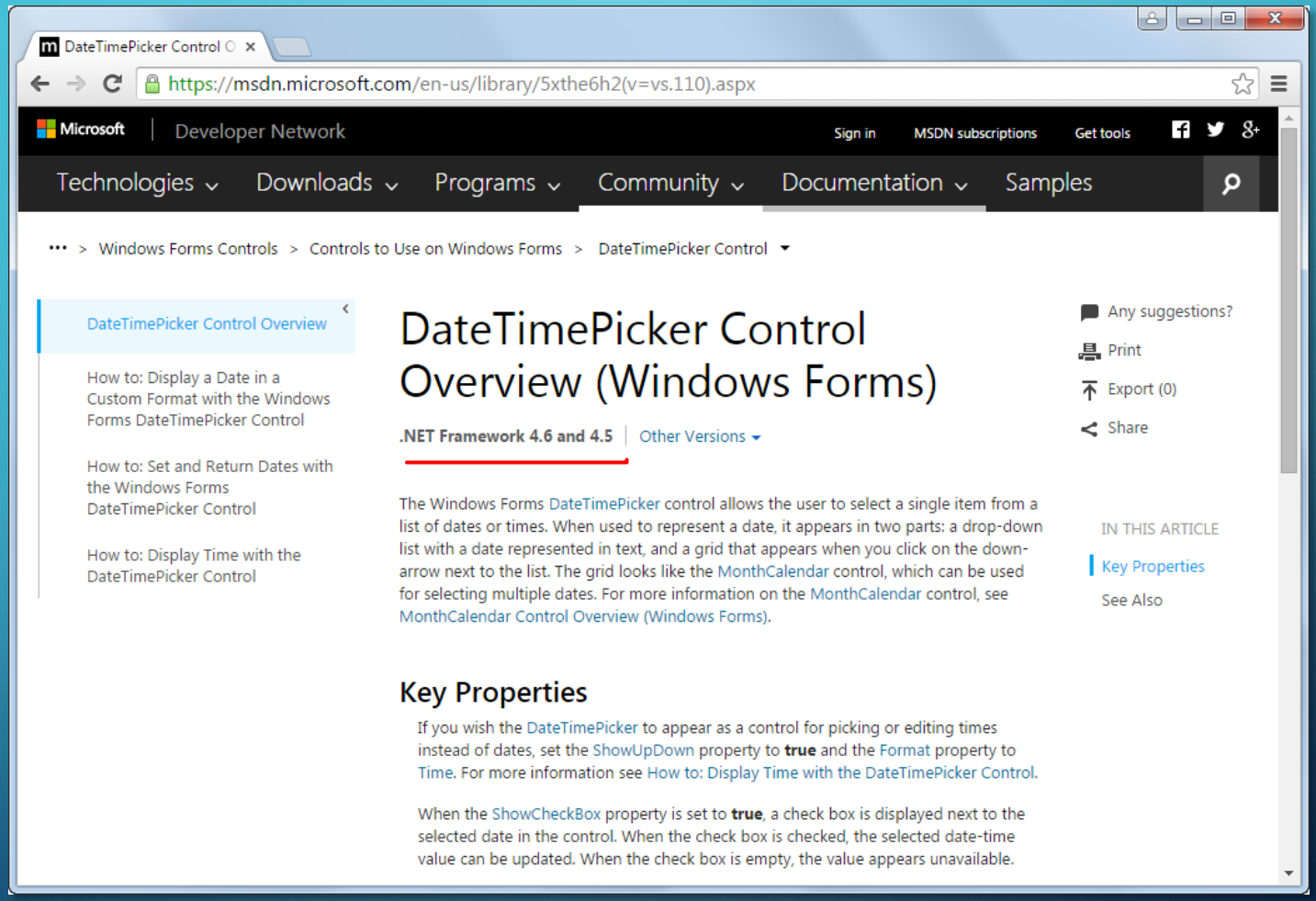




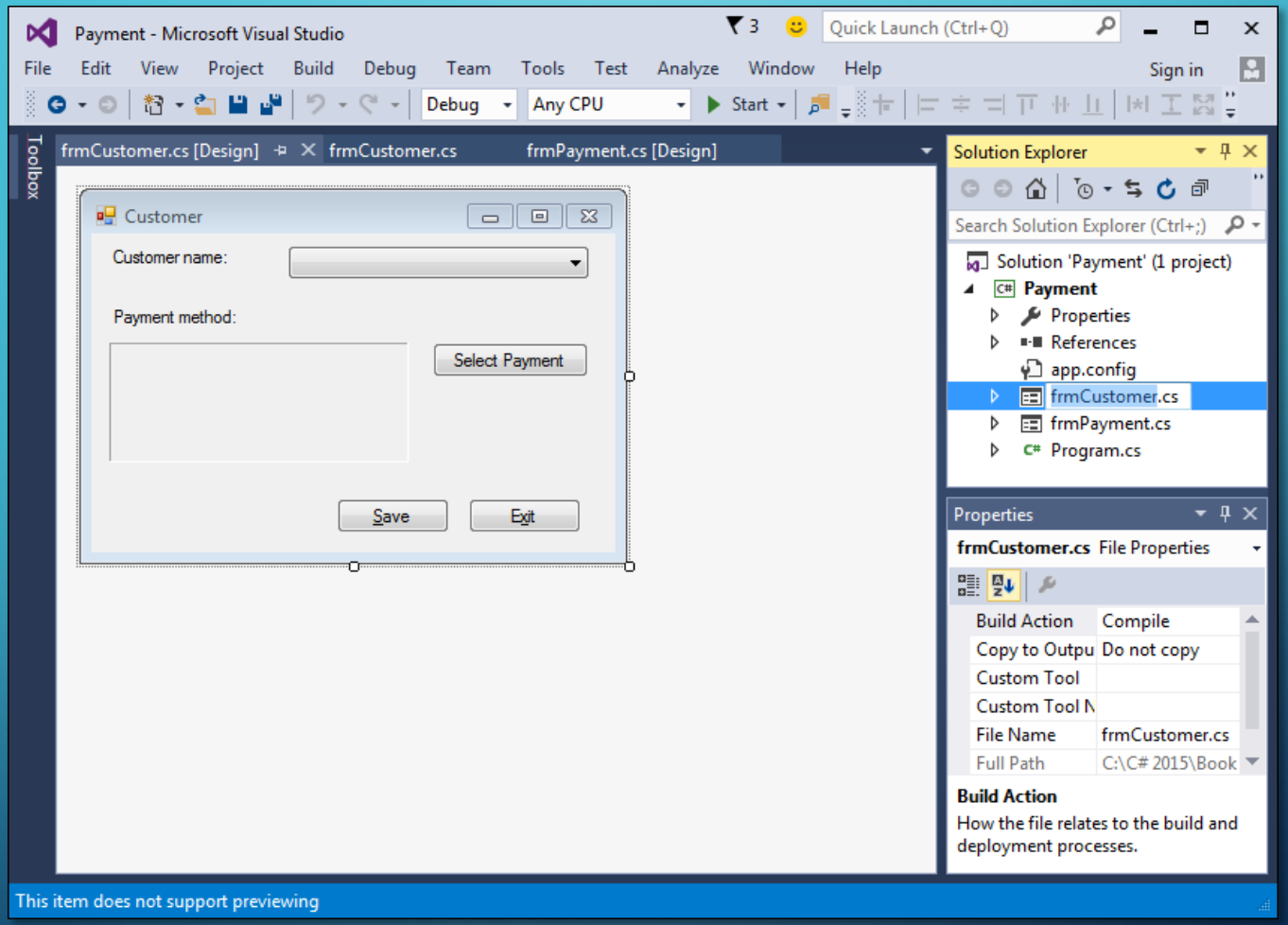
HOW TO USE TAB ORDER VIEW

- To display a form in Tab Order view, select the form and then select the View→Tab Order command. This displays the tab index for each control.
- To change the tab indexes of the controls, click on the controls in the sequence you want to use. As you click, the new tab indexes appear.
- The first time you click on a control in Tab Order view, the tab index is set to the next number in sequence, starting with zero for the first control. If you click on the same control more than once, the tab index is increased by one each time you click.
- If a group box contains other controls, the controls in the group box are displayed with sub indexes. Then, you can click on the group box to change its index and the main indexes of the controls it contains. To change the sub indexes of the controls in the group box, click on them individually.

HELP FOR THE DATETIMEPICKER CONTROL



A PROJECT THAT CONTAINS TWO FORMS



HOW TO CHANGE THE NAME OF A FORM

1. Right-click the form in the Solution Explorer and select the Rename command. Or, select the form in the Solution Explorer and press F2.
2. Enter the new name for the form. When you do, Visual Studio asks if you want to rename all references to the file. In most cases, that's what you'll want to do.

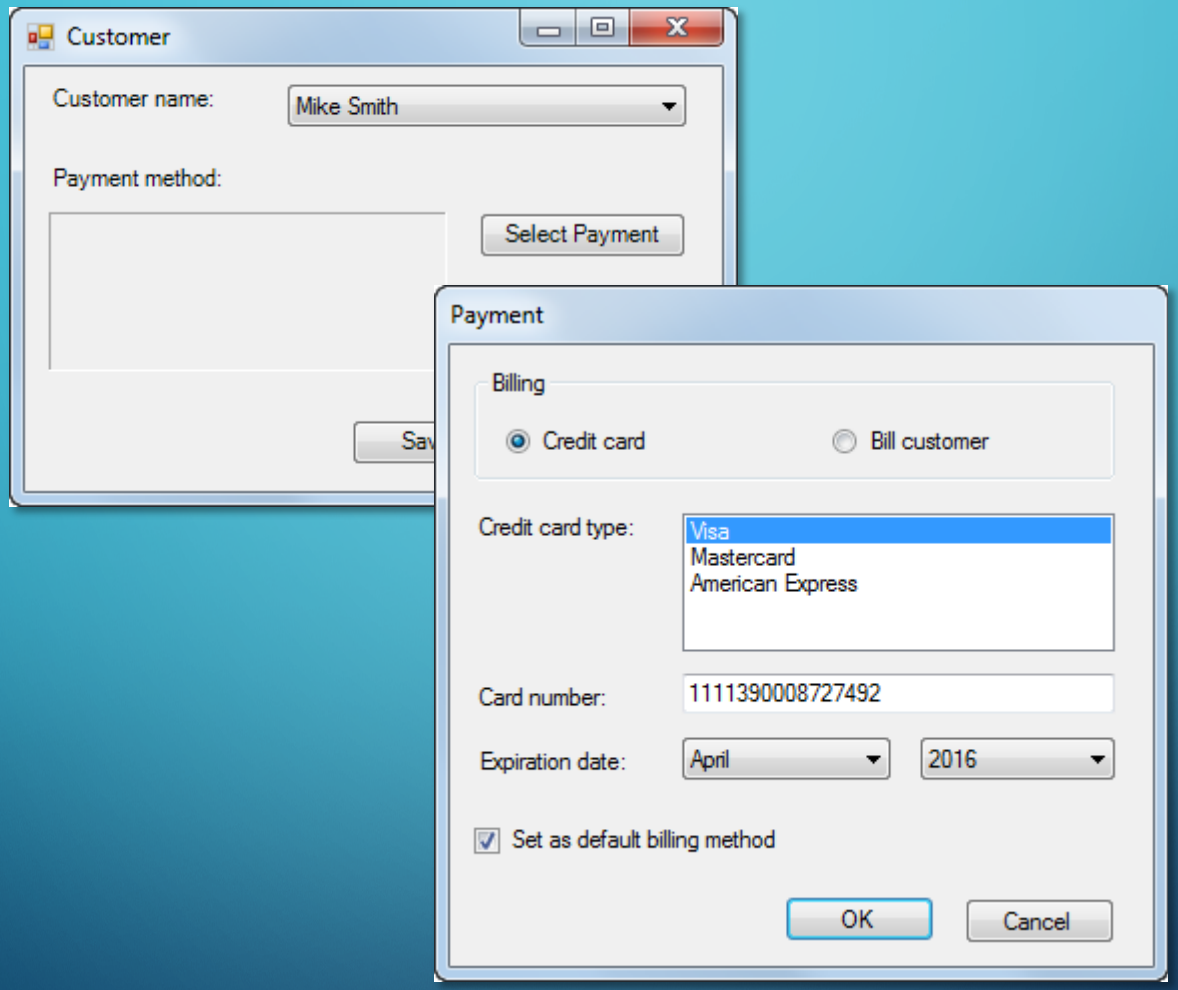
How to change the name of any event handlers for a form's events

1. Highlight the name of the form in the method that's used by the event handler, then right-click on the name and choose Rename from the shortcut menu that's displayed.
2. Enter the new name for the form and click the Apply button in the Rename dialog box.

CODE THAT DEFINES THE MAIN ENTRY POINT FOR AN APPLICATION

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Payment
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new frmCustomer());
        }
    }
}
```

THE PAYMENT FORM DISPLAYED AS A DIALOG BOX



PROPERTIES FOR CREATING CUSTOM DIALOG BOXES

- `FormBorderStyle`
- `ControlBox`
- `MaximizeBox`
- `MinimizeBox`

CODE THAT CREATES AND DISPLAYS A CUSTOM DIALOG BOX

```
Form paymentForm = new frmPayment();
paymentForm.ShowDialog();
// execution continues here after the user responds
// to the dialog box
```

AN ENUMERATION THAT WORKS WITH DIALOG BOXES

Enumeration	Members
`DialogResult`	OK, Cancel, Yes, No, Abort, Retry, Ignore, None

The Tag property

Property	Description
`Tag`	Gets or sets data associated with the form or a control. The Tag property holds a reference to an object type, which means that it can hold any type of data.

A statement that sets the DialogResult property of a form

```
this.DialogResult = DialogResult.OK;
```

A statement that sets the Tag property of a form

```
this.Tag = msg;
```

Code that uses the result of a dialog box and the Tag property

```
Form paymentForm = new frmPayment();
DialogResult selectedButton = paymentForm.ShowDialog();
if (selectedButton == DialogResult.OK)
    lblPayment.Text = paymentForm.Tag.ToString();
```

HOW TO USE THE DIALOGRESULT ENUMERATION

- The DialogResult enumeration provides members that represent the values that a dialog box can return. The ShowDialog method returns a member of this enumeration.
- You specify the result value of a custom dialog box by setting its DialogResult property. Or, you can set the DialogResult property of a button in the dialog box. Then, when the user clicks that button, the DialogResult property of the form is set accordingly.
- If you set the CancelButton property of a form to a button on that form, the DialogResult property of that button is automatically set to Cancel.
- After you set the DialogResult property of a dialog box, the form is closed and control is returned to the form that displayed it. If you close a dialog box without setting the DialogResult property, a value of Cancel is returned to the main form.

HOW TO USE THE TAG PROPERTY

- The Tag property provides a convenient way to pass data between forms in a multi-form application.
- A dialog box can set its Tag property before it returns control to the main form. Then, the main form can get the data from this property and use it as necessary.
- Because the Tag property is an object type, you must explicitly cast it to the appropriate type to retrieve the data it contains. Or, you can use the ToString method to convert the data to a string.

THE SYNTAX FOR THE SHOW METHOD OF THE MESSAGEBOX CLASS

```
MessageBox.Show(text[, caption[, buttons[, icon
    [, defaultButton]]]]);
```

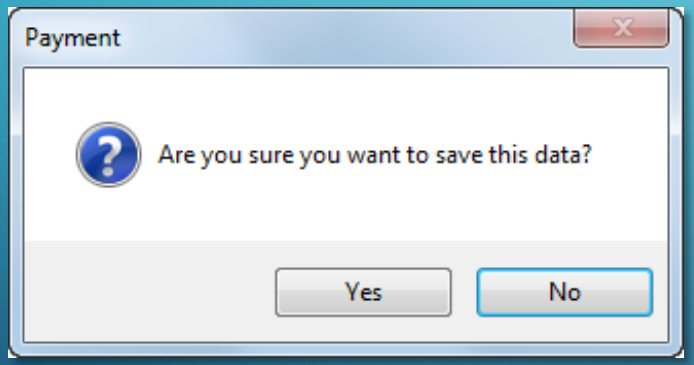
The enumerations that work with the MessageBox class

Enumeration	Members
`MessageBoxButtons`	OK, OKCancel, YesNo, YesNoCancel, AbortRetryIgnore
`MessageBoxIcon`	None, Information, Error, Warning, Exclamation, Question, Asterisk, Hand, Stop
`MessageBoxDefaultButton`	Button1, Button2, Button3
`DialogResult`	OK, Cancel, Yes, No, Abort, Retry, Ignore

A STATEMENT THAT DISPLAYS A DIALOG BOX AND GETS THE USER RESPONSE

```
DialogResult button =
    MessageBox.Show(
        "Are you sure you want to save this data?",
        "Payment",
        MessageBoxButtons.YesNo,
        MessageBoxIcon.Question,
        MessageBoxDefaultButton.Button2);
```

The dialog box that's displayed



A STATEMENT THAT CHECKS THE USER RESPONSE

```
if (button == DialogResult.Yes)
{
    SaveData();
    isDataSaved = true;
}
```

THE CODE FOR A DIALOG BOX THAT CANCELS THE CLOSING EVENT

```
private void frmCustomer_FormClosing(object sender, FormClosingEventArgs e)
{
    if (isDataSaved == false)
    {
        string message =
            "This form contains unsaved data.\n\n" +
            "Do you want to save it?";

        DialogResult button =
            MessageBox.Show(message, "Customer",
            MessageBoxButtons.YesNoCancel,
            MessageBoxIcon.Warning);

        if (button == DialogResult.Yes)
        {
            if (IsValidData())
                this.SaveData();
            else
                e.Cancel = true;
        }
        if (button == DialogResult.Cancel)
        {
            e.Cancel = true;
        }
    }
}
```

THE DIALOG BOX THAT'S DISPLAYED