Chapter 3

How to code and test a Windows Forms application

Objectives

Applied

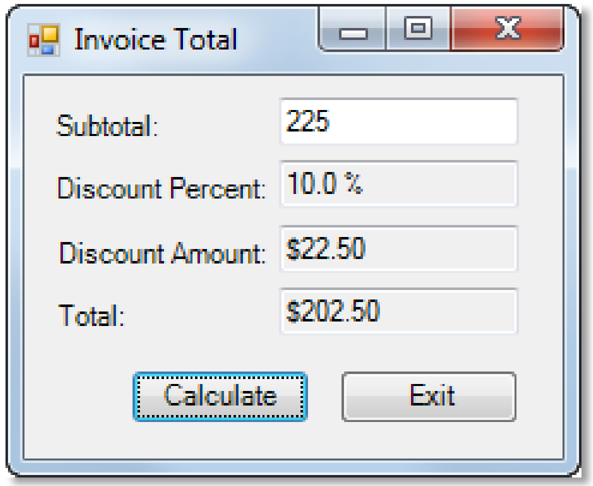
1. Given the code for a simple application, use the skills presented in this chapter to add the code and test the application.
2. Use indentation and blank lines to make the code easier to read.
3. Use comments to document the code for the entire form or for portions of the code.
4. Use any of the help features to get the information that you need for developing an application.

Objectives (cont.)

Knowledge

1. In the context of object-oriented programming, describe these terms: class, object, instantiation, instance, property, method, event, and member.
2. Describe how an application responds to events.
3. Describe the use of snippets, refactoring, and annotations.
4. Describe the use of annotations in the vertical scroll bar of the Code Editor.
5. Distinguish between a syntax (or build) error and a runtime error.
6. Distinguish between testing and debugging.
7. Explain how a data tip can help debug a runtime error.

A form object and its ten control objects


Invoice Total

Subtotal:	225
Discount Percent:	10.0 %
Discount Amount:	$22.50
Total:	$202.50

Class and object concepts

- An *object* is a self-contained unit like a form or control that combines code and data.
- A *class* is the code that defines the characteristics of an object. You can think of a class as a template for an object.
- An object is an *instance* of a class, and the process of creating an object from a class is called *instantiation*.
- More than one object instance can be created from a single class.

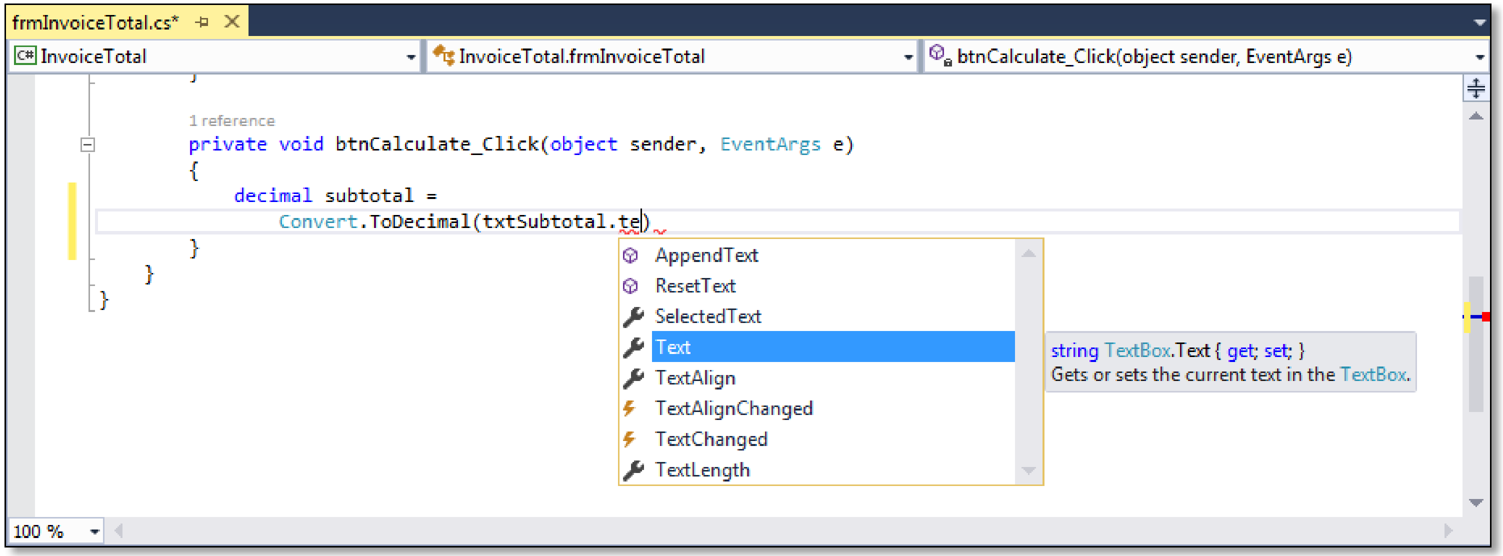
Property, method, and event concepts

- *Properties* define the characteristics of an object and the data associated with an object.
- *Methods* are the operations that an object can perform.
- *Events* are signals sent by an object to the application telling it that something has happened that can be responded to.
- Properties, methods, and events can be referred to as *members* of an object.
- If you instantiate two or more instances of the same class, all of the objects have the same properties, methods, and events. However, the values assigned to the properties can vary from one instance to another.

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.

A member list in the Code Editor window


frmInvoiceTotal.cs*
InvoiceTotal
InvoiceTotal.frmInvoiceTotal
btnCalculate_Click(object sender, EventArgs e)
100 %

```
        1 reference
        private void btnCalculate_Click(object sender, EventArgs e)
        {
            decimal subtotal =
                Convert.ToDecimal(txtSubtotal.te)
        }
    }
}
```

string TextBox.Text { get; set; }
Gets or sets the current text in the TextBox.

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.

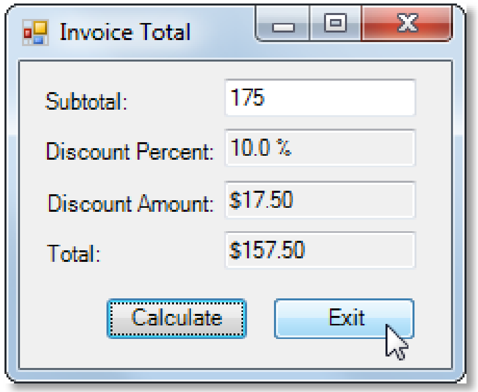
How to enter member names when working in the Code Editor

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



Subtotal:	175
Discount Percent:	10.0 %
Discount Amount:	$17.50
Total:	$157.50

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.

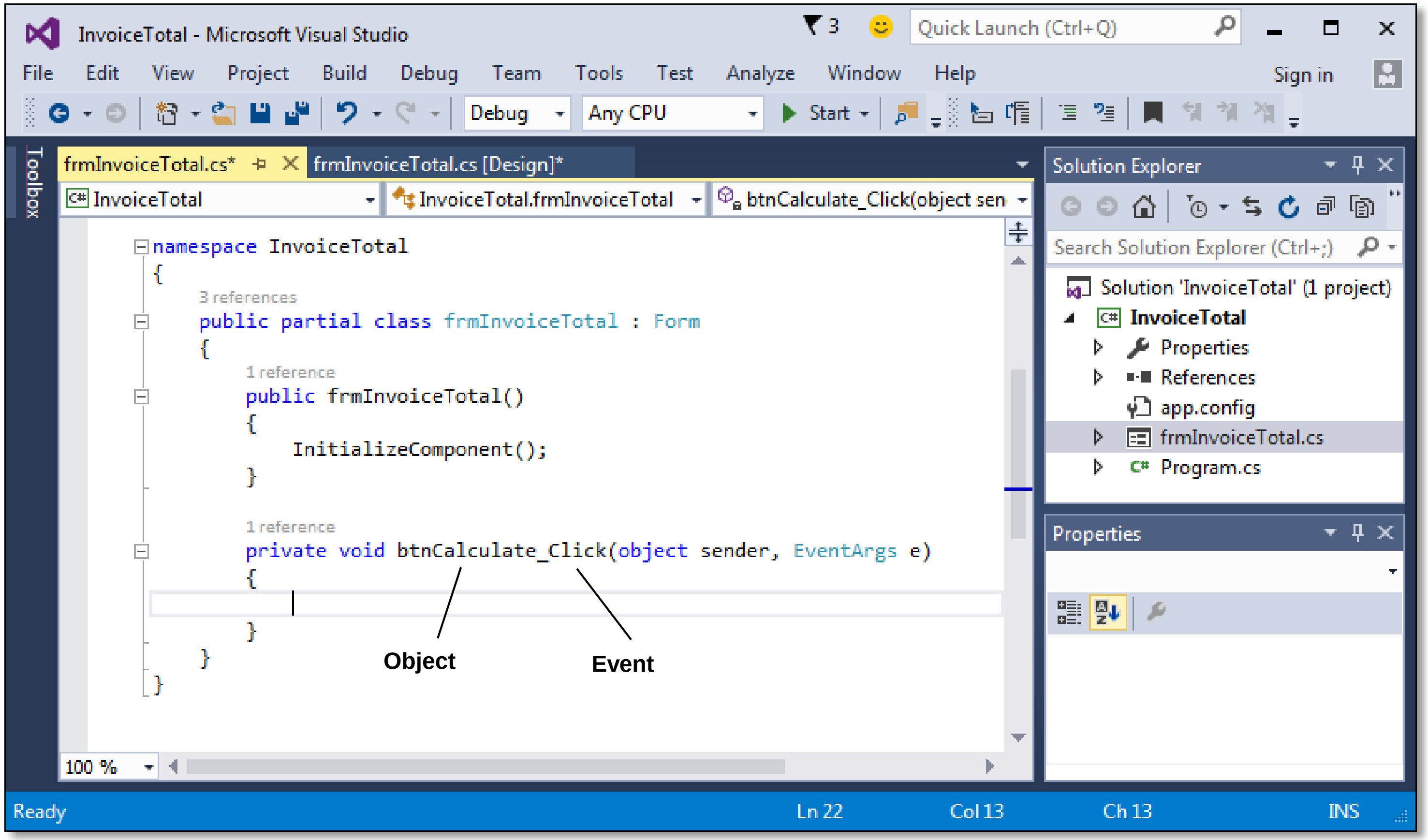
Common form events

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



InvoiceTotal - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug
Any CPU
Start
frmInvoiceTotal.cs*
frmInvoiceTotal.cs [Design]*
InvoiceTotal
InvoiceTotal.frmInvoiceTotal
btnCalculate_Click(object sen
Solution Explorer
Solution 'InvoiceTotal' (1 project)
InvoiceTotal
Properties
References
app.config
frmInvoiceTotal.cs
Program.cs
Properties
Ready
Ln 22
Col 13
Ch 13
INS
Object
Event

```
namespace InvoiceTotal
{
    3 references
    public partial class frmInvoiceTotal : Form
    {
        1 reference
        public frmInvoiceTotal()
        {
            InitializeComponent();
        }

        1 reference
        private void btnCalculate_Click(object sender, EventArgs e)
        {

        }
    }
}
```

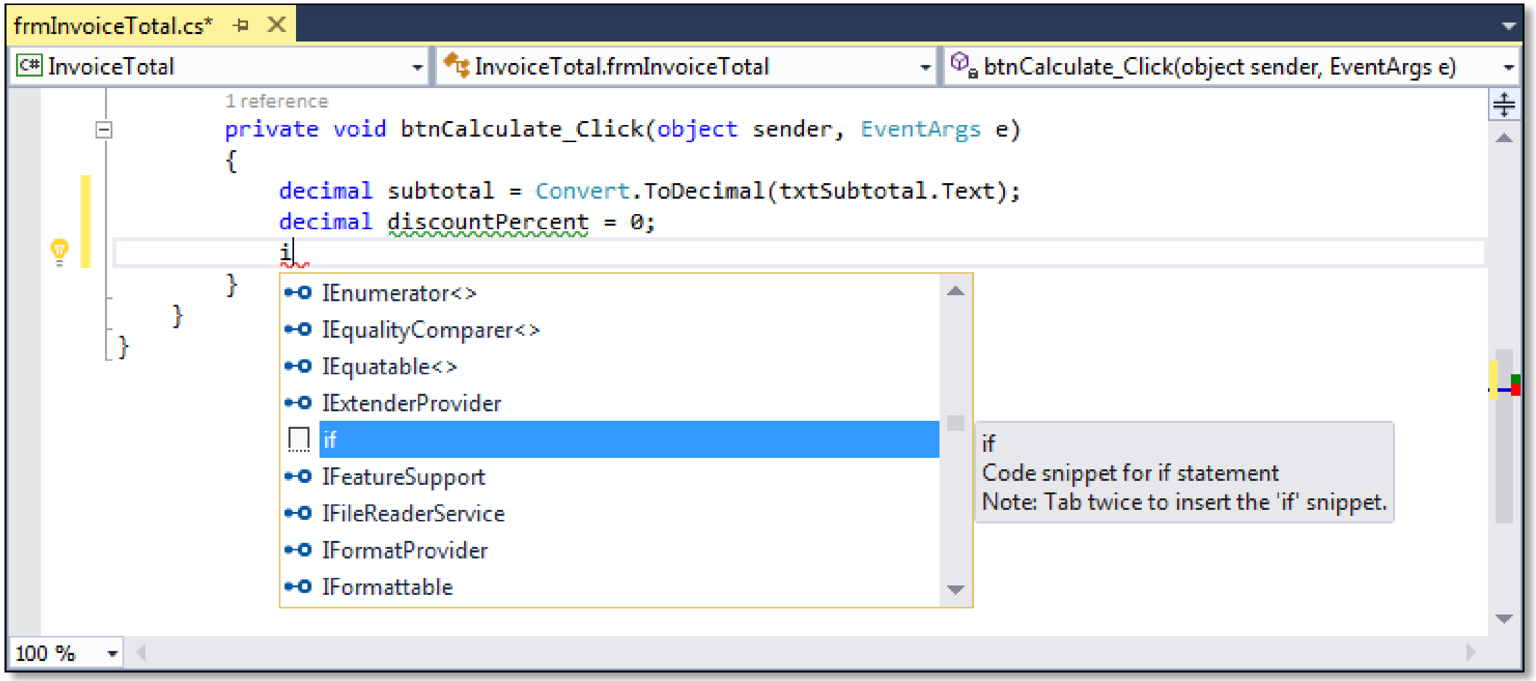
How to handle the Click event of a button

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 completion list that's displayed when you enter a letter at the beginning of a line of code

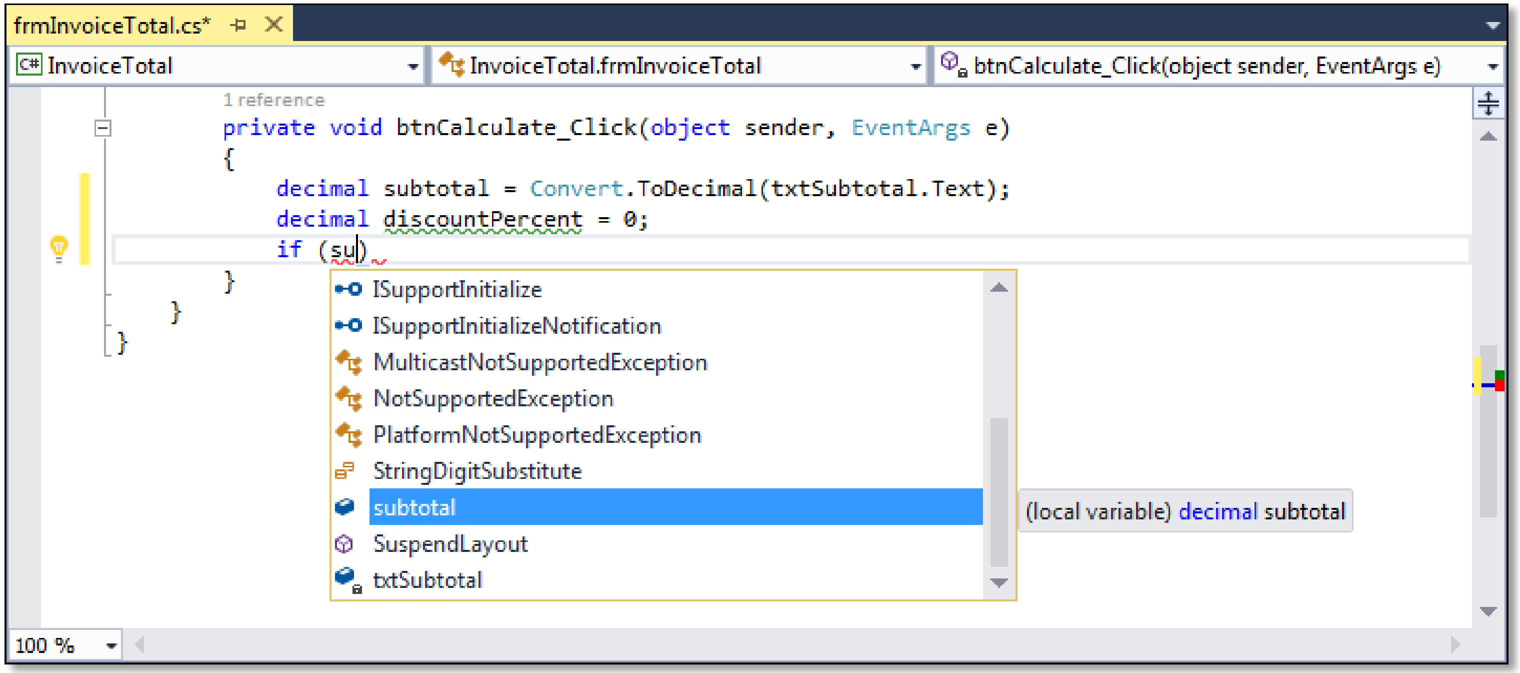

frmInvoiceTotal.cs*
InvoiceTotal
InvoiceTotal.frmInvoiceTotal
btnCalculate_Click(object sender, EventArgs e)
100 %

```
        1 reference
        private void btnCalculate_Click(object sender, EventArgs e)
        {
            decimal subtotal = Convert.ToDecimal(txtSubtotal.Text);
            decimal discountPercent = 0;
            i
        }
    }
}
```

```
IEnumerator<>
IEqualityComparer<>
IEquatable<>
IExtenderProvider
if
IFeatureSupport
IFileReaderService
IFormatProvider
IFormattable
```

if
Code snippet for if statement
Note: Tab twice to insert the 'if' snippet.

The completion list that's displayed as you enter code within a statement



frmInvoiceTotal.cs*

InvoiceTotal | InvoiceTotal.frmInvoiceTotal | btnCalculate_Click(object sender, EventArgs e)

```
        1 reference
        private void btnCalculate_Click(object sender, EventArgs e)
        {
            decimal subtotal = Convert.ToDecimal(txtSubtotal.Text);
            decimal discountPercent = 0;
            if (su)
        }
    }
}
```

ISupportInitialize
ISupportInitializeNotification
MulticastNotSupportedException
NotSupportedException
PlatformNotSupportedException
StringDigitSubstitute
subtotal
SuspendLayout
txtSubtotal

(local variable) decimal subtotal

100 %

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;
```

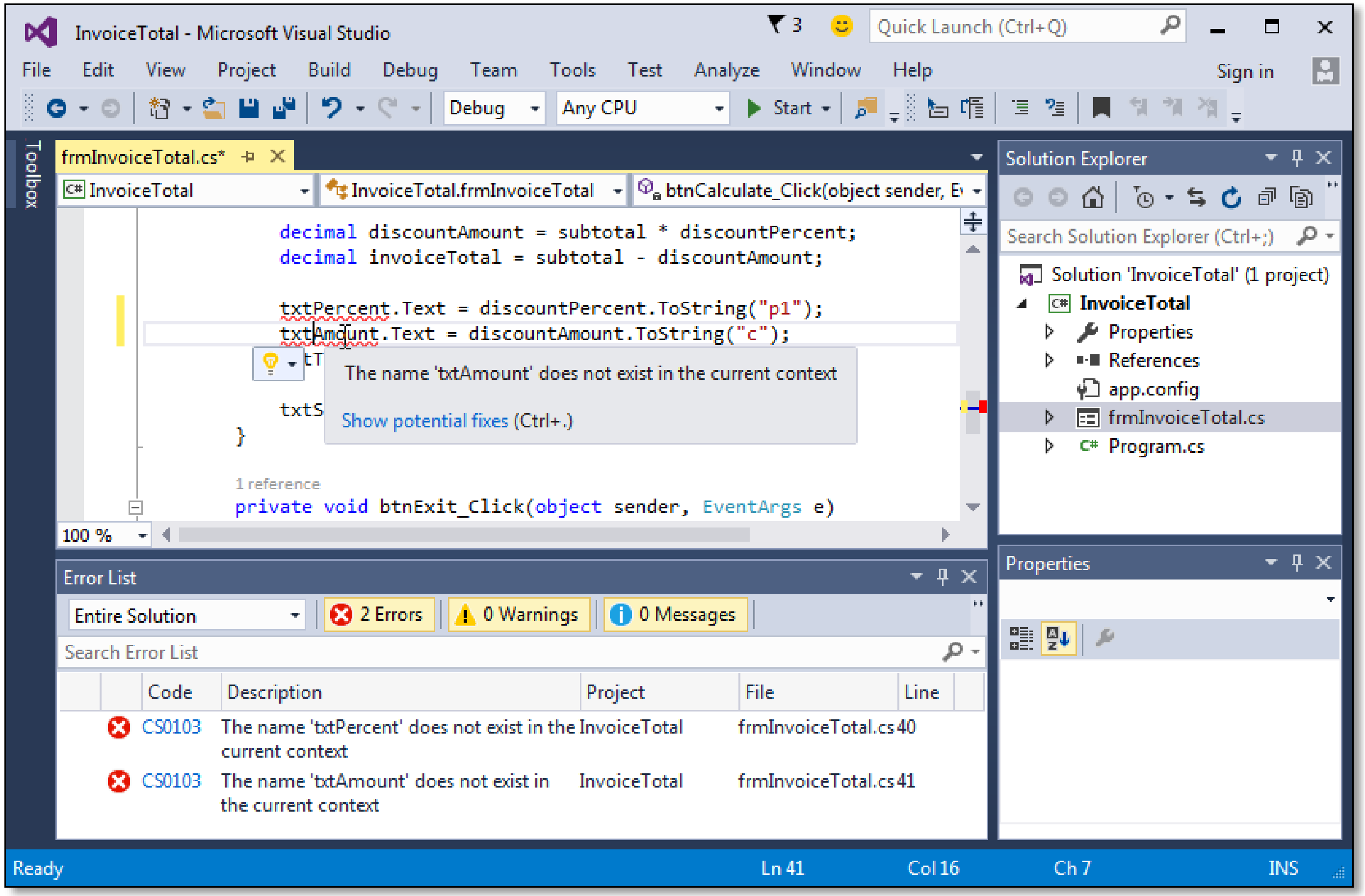
The event handlers for the Invoice Total form (cont.)

```
        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();
    }
```

Coding rules

- Use spaces to separate the words in each statement.
- Use exact capitalization for all keywords, class names, object names, variable names, etc.
- End each *statement* with a semicolon.
- Each *block* of code must be enclosed in braces ({}). That includes the block of code that defines the body of a method.

The Code Editor and Error List windows with syntax errors displayed



InvoiceTotal - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Sign in
Quick Launch (Ctrl+Q)
Debug
Any CPU
Start
frmInvoiceTotal.cs*
InvoiceTotal
InvoiceTotal.frmInvoiceTotal
btnCalculate_Click(object sender, E
Toolbox
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'InvoiceTotal' (1 project)
InvoiceTotal
Properties
References
app.config
frmInvoiceTotal.cs
Program.cs
Properties
100 %
Ready
Ln 41
Col 16
Ch 7
INS

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

            txtPercent.Text = discountPercent.ToString("p1");
            txtAmount.Text = discountAmount.ToString("c");
            tT
            
            txtS
        }

        1 reference
        private void btnExit_Click(object sender, EventArgs e)
```

The name 'txtAmount' does not exist in the current context

Show potential fixes (Ctrl+.)

Error List

Entire Solution | 2 Errors | 0 Warnings | 0 Messages

Search Error List

Code	Description	Project	File	Line
CS0103	The name 'txtPercent' does not exist in the current context	InvoiceTotal	frmInvoiceTotal.cs	40
CS0103	The name 'txtAmount' does not exist in the current context	InvoiceTotal	frmInvoiceTotal.cs	41

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;
    }
```

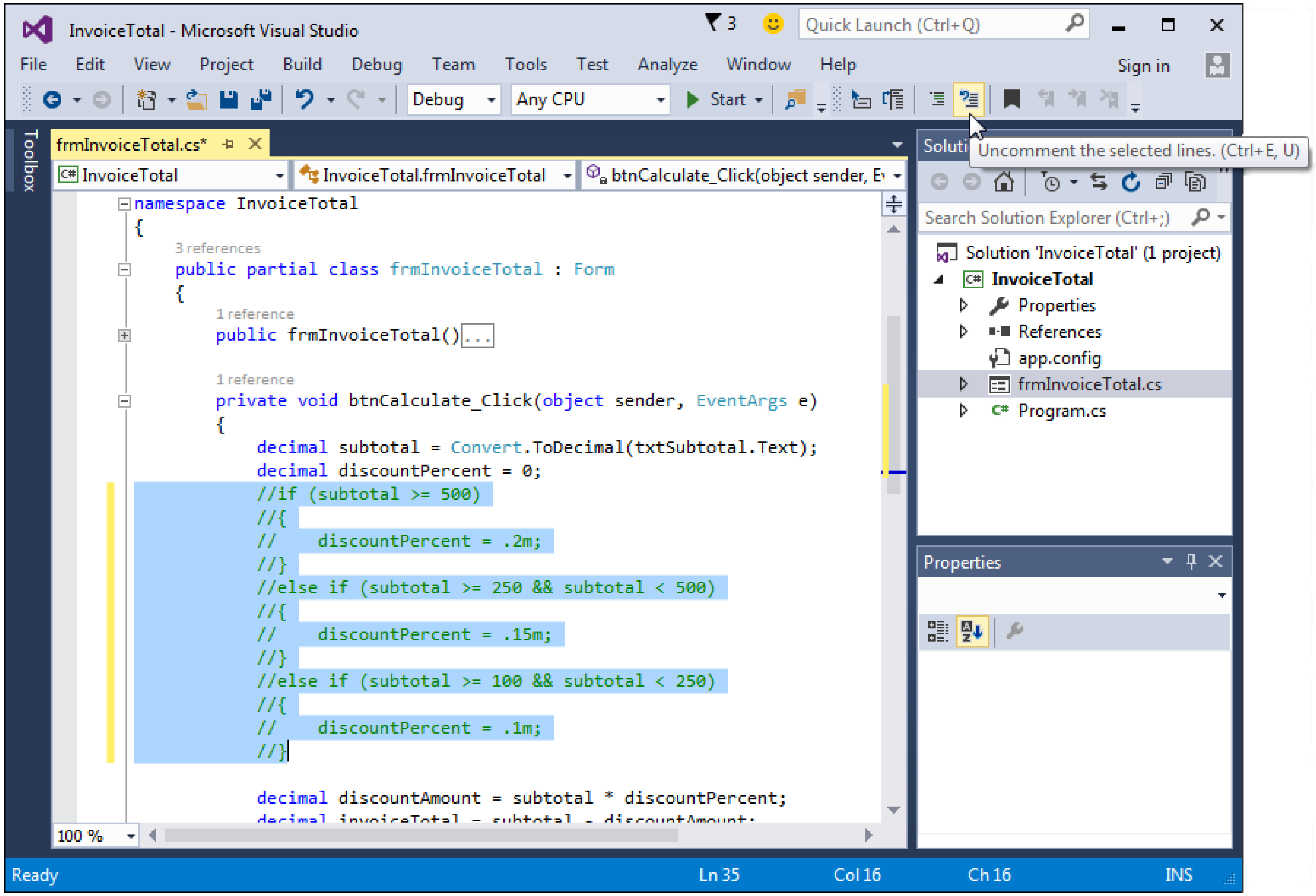
A method with comments (cont.)

```
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


InvoiceTotal - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Sign in
Debug
Any CPU
Start
Uncomment the selected lines. (Ctrl+E, U)
frmInvoiceTotal.cs*
InvoiceTotal
InvoiceTotal.frmInvoiceTotal
btnCalculate_Click(object sender, E
Search Solution Explorer (Ctrl+;)
Solution 'InvoiceTotal' (1 project)
InvoiceTotal
Properties
References
app.config
frmInvoiceTotal.cs
Program.cs
Properties
100 %
Ready
Ln 35
Col 16
Ch 16
INS

```
namespace InvoiceTotal
{
    3 references
    public partial class frmInvoiceTotal : Form
    {
        1 reference
        public frmInvoiceTotal()...

        1 reference
        private void btnCalculate_Click(object sender, EventArgs e)
        {
            decimal subtotal = Convert.ToDecimal(txtSubtotal.Text);
            decimal discountPercent = 0;
            //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;
```

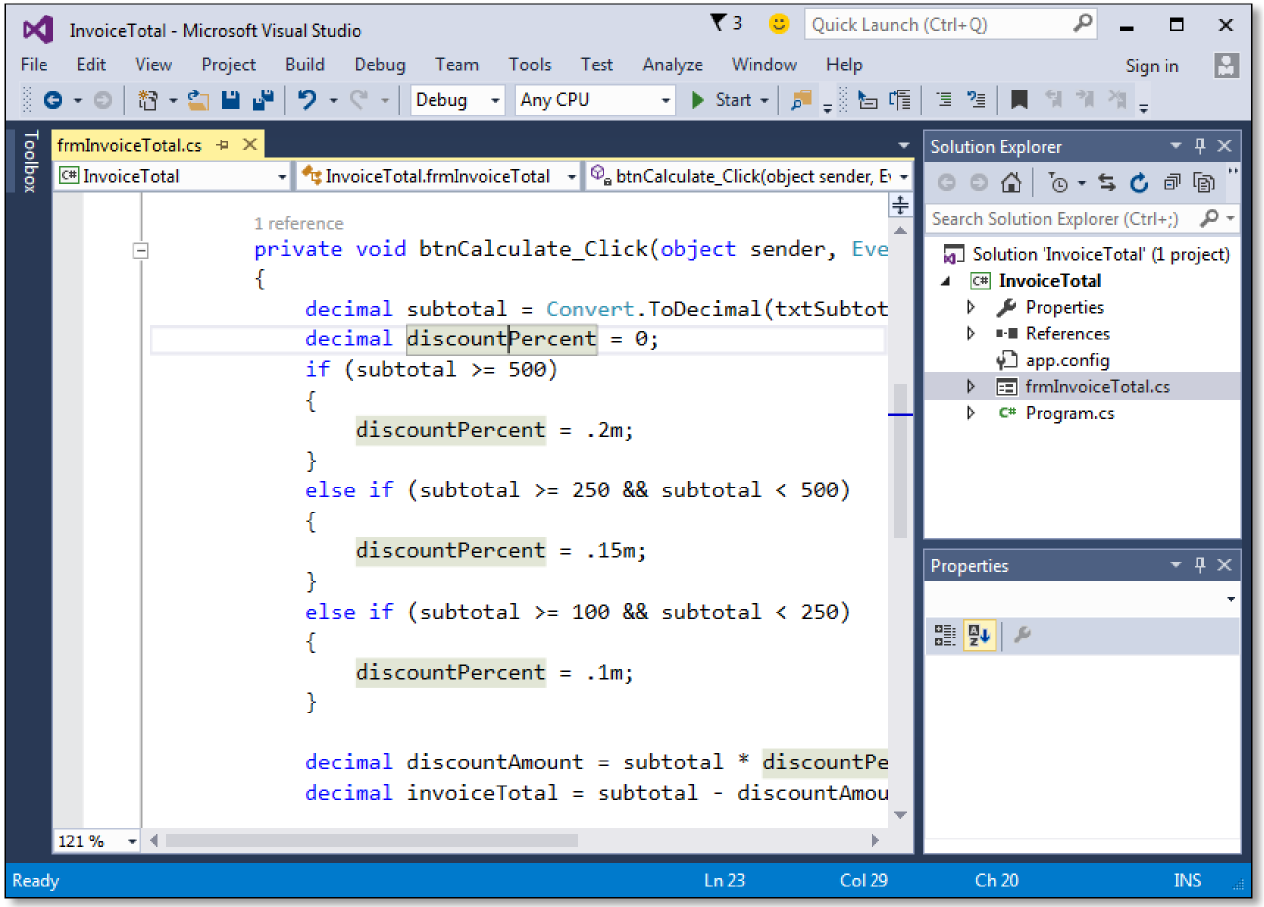
How to use the buttons of the Text Editor toolbar

- 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.

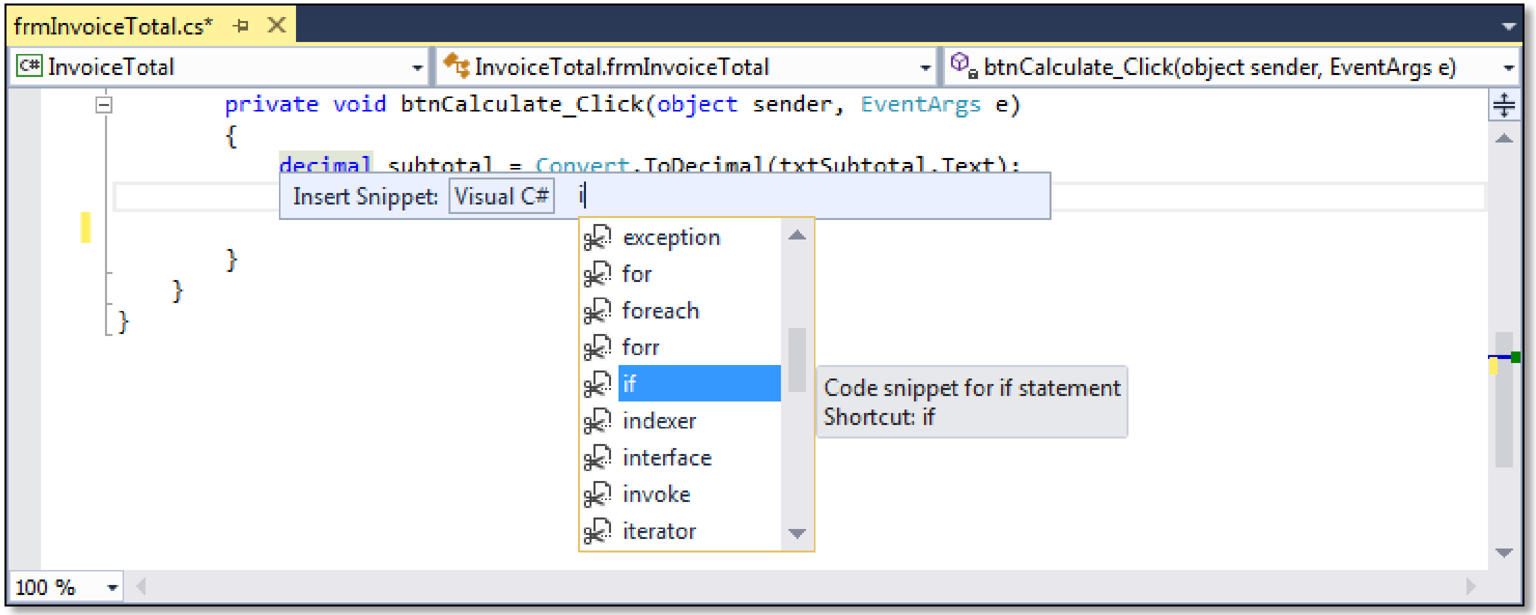
The Code Editor with enlarged text and a highlighted variable



InvoiceTotal - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Sign in
Debug
Any CPU
Start
Toolbox
frmInvoiceTotal.cs
InvoiceTotal
InvoiceTotal.frmInvoiceTotal
btnCalculate_Click(object sender, E
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'InvoiceTotal' (1 project)
InvoiceTotal
Properties
References
app.config
frmInvoiceTotal.cs
Program.cs
Properties
121 %
Ready
Ln 23
Col 29
Ch 20
INS

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

            decimal discountAmount = subtotal * discountPe
            decimal invoiceTotal = subtotal - discountAmou
```

The default list of Visual C# code snippets



frmInvoiceTotal.cs*
InvoiceTotal
InvoiceTotal.frmInvoiceTotal
btnCalculate_Click(object sender, EventArgs e)

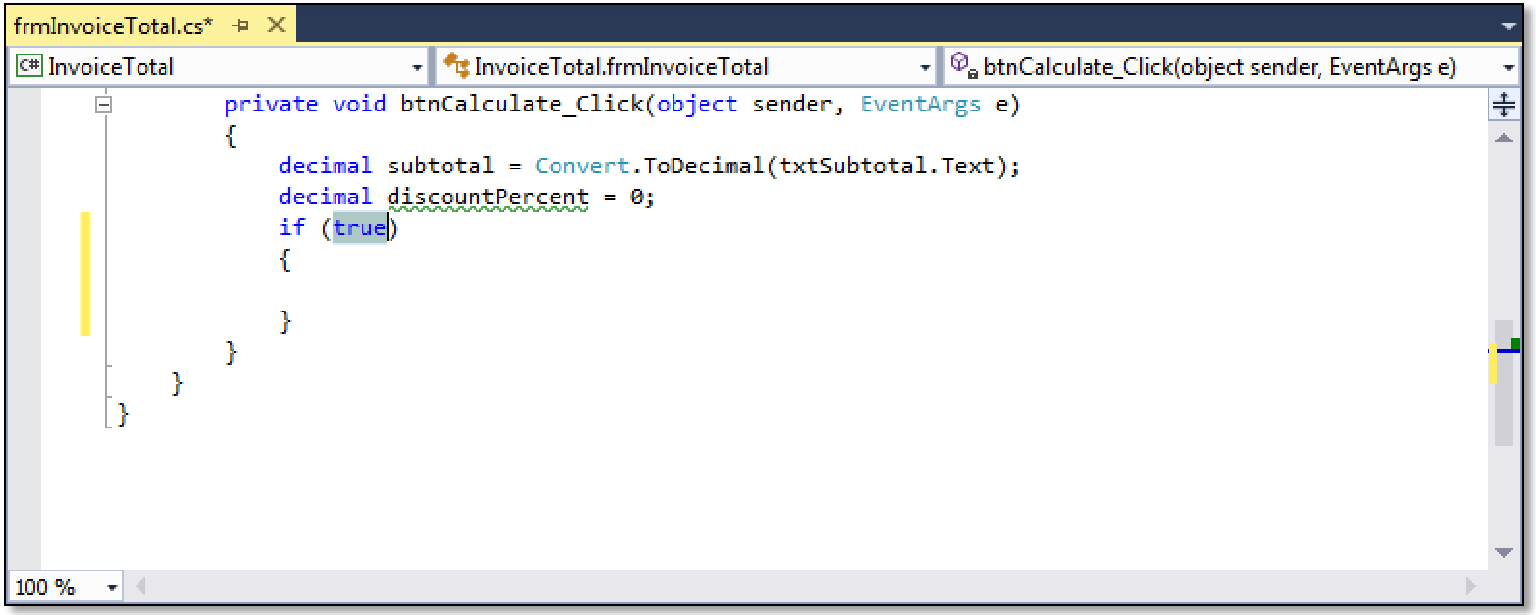
```
        private void btnCalculate_Click(object sender, EventArgs e)
        {
            decimal subtotal = Convert.ToDecimal(txtSubtotal.Text);
```

Insert Snippet: Visual C# i

Code snippet for if statement
Shortcut: if

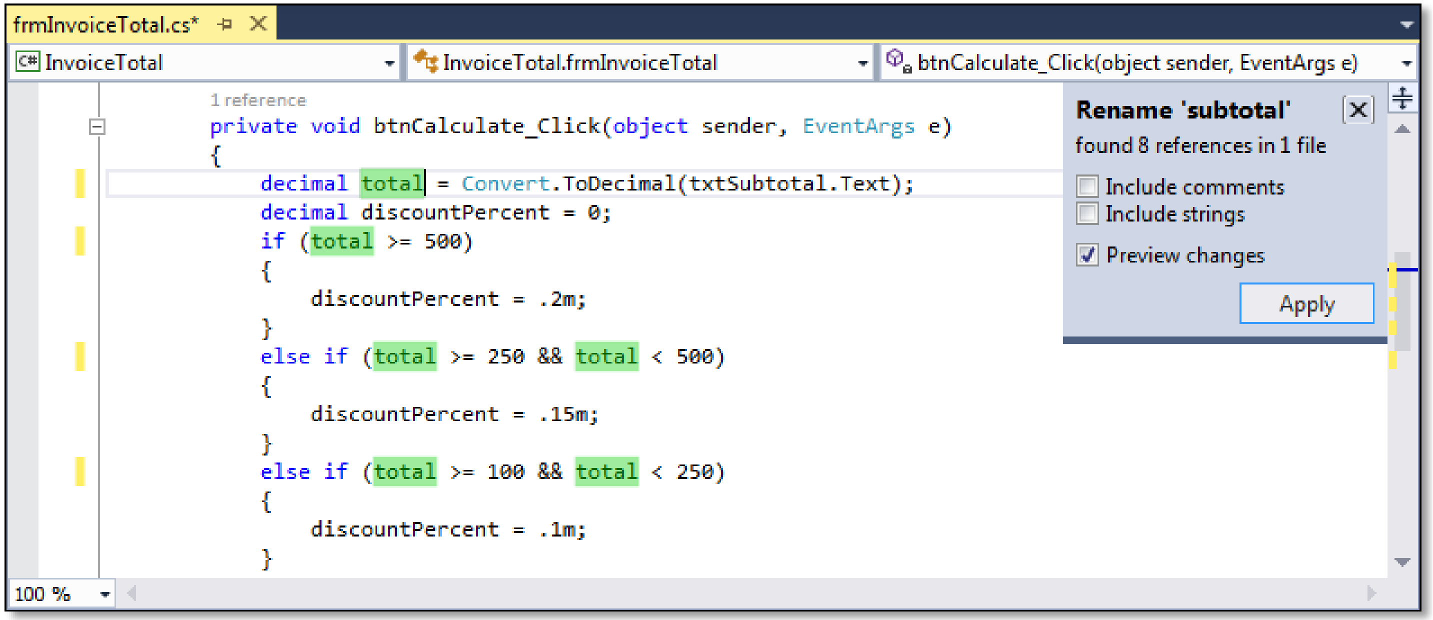
```
        }
    }
}
```

A code snippet after it has been inserted


frmInvoiceTotal.cs*
InvoiceTotal
InvoiceTotal.frmInvoiceTotal
btnCalculate_Click(object sender, EventArgs e)
100 %

```
        private void btnCalculate_Click(object sender, EventArgs e)
        {
            decimal subtotal = Convert.ToDecimal(txtSubtotal.Text);
            decimal discountPercent = 0;
            if (true)
            {

            }
        }
    }
}
```

The options that are displayed when you rename a variable



frmInvoiceTotal.cs*
InvoiceTotal
InvoiceTotal.frmInvoiceTotal
btnCalculate_Click(object sender, EventArgs e)
1 reference
100 %

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

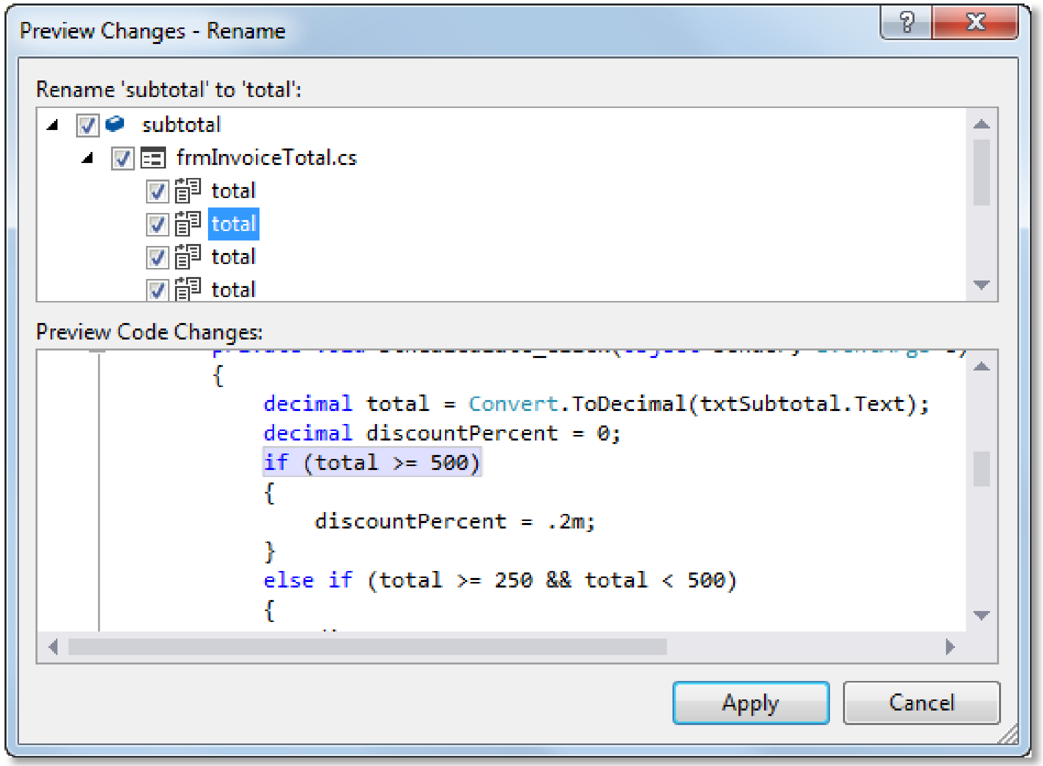
Rename 'subtotal'

found 8 references in 1 file

- [ ] Include comments
- [ ] Include strings
- [✓] Preview changes

Apply

The Preview Changes - Rename dialog box



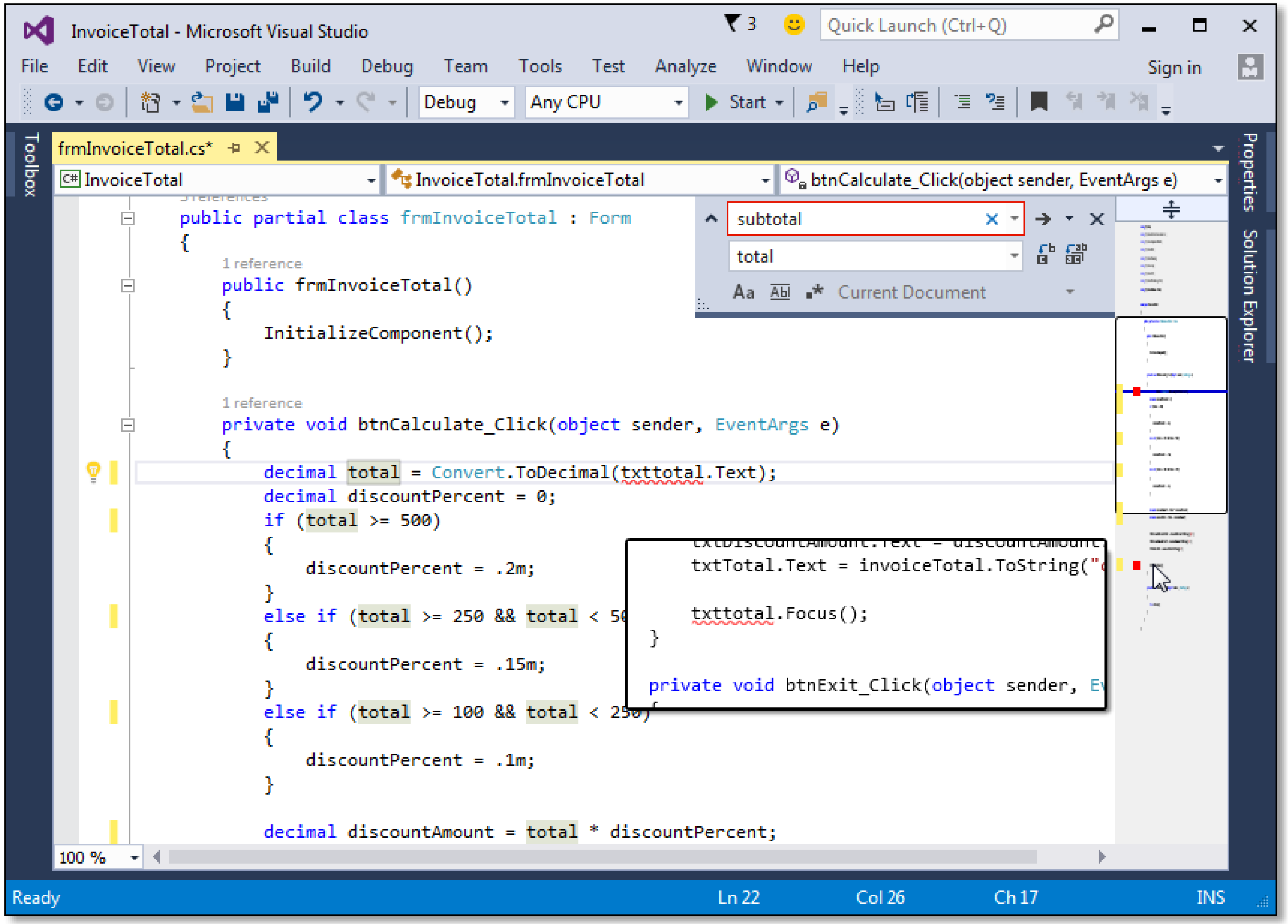
Rename 'subtotal' to 'total':

- subtotal
  - frmInvoiceTotal.cs
    - ✓ total
    - ✓ total
    - ✓ total
    - ✓ total

Preview Code Changes:

```
{
    decimal total = Convert.ToDecimal(txtSubtotal.Text);
    decimal discountPercent = 0;
    if (total >= 500)
    {
        discountPercent = .2m;
    }
    else if (total >= 250 && total < 500)
    {
```

A scroll bar in map mode with annotations


InvoiceTotal - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Test Analyze Window Help

frmInvoiceTotal.cs*

InvoiceTotal | InvoiceTotal.frmInvoiceTotal | btnCalculate_Click(object sender, EventArgs e)

Find: subtotal — Replace: total — Current Document

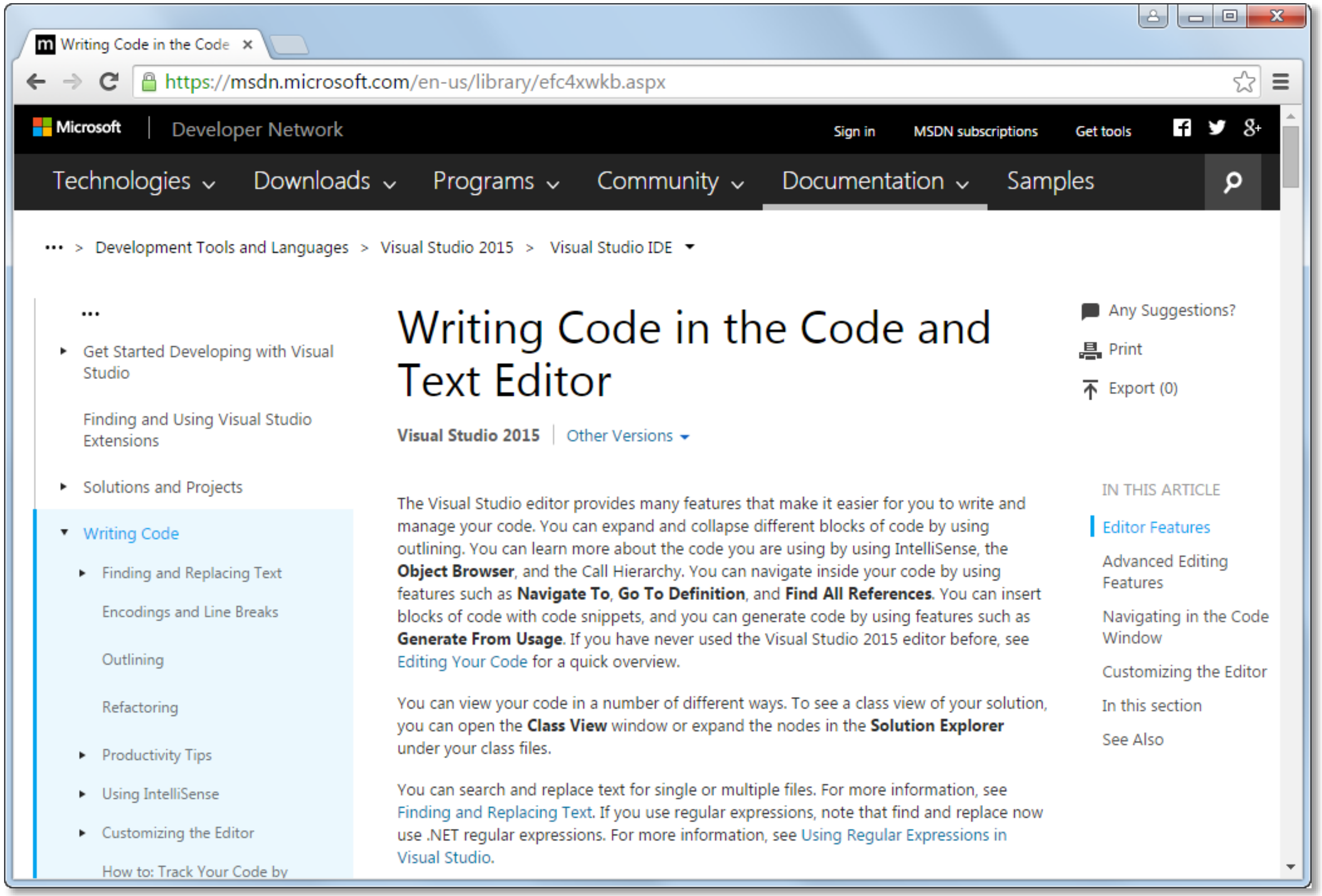
```
public partial class frmInvoiceTotal : Form
{
    1 reference
    public frmInvoiceTotal()
    {
        InitializeComponent();
    }

    1 reference
    private void btnCalculate_Click(object sender, EventArgs e)
    {
        decimal total = Convert.ToDecimal(txttotal.Text);
        decimal discountPercent = 0;
        if (total >= 500)
        {
            discountPercent = .2m;
        }
        else if (total >= 250 && total < 50
        {
            discountPercent = .15m;
        }
        else if (total >= 100 && total < 250)
        {
            discountPercent = .1m;
        }

        decimal discountAmount = total * discountPercent;
```

```
    txtTotal.Text = invoiceTotal.ToString("
    
    txttotal.Focus();
}

private void btnExit_Click(object sender, E
```

100 %

Ready Ln 22 Col 26 Ch 17 INS

Online help for Visual Studio


Writing Code in the Code

https://msdn.microsoft.com/en-us/library/efc4xwkb.aspx

Microsoft | Developer Network

Sign in MSDN subscriptions Get tools

Technologies Downloads Programs Community Documentation Samples

... > Development Tools and Languages > Visual Studio 2015 > Visual Studio IDE

...
- Get Started Developing with Visual Studio
- Finding and Using Visual Studio Extensions
- Solutions and Projects
- Writing Code
  - Finding and Replacing Text
  - Encodings and Line Breaks
  - Outlining
  - Refactoring
  - Productivity Tips
  - Using IntelliSense
  - Customizing the Editor
  - How to: Track Your Code by

Writing Code in the Code and Text Editor

Visual Studio 2015 | Other Versions

The Visual Studio editor provides many features that make it easier for you to write and manage your code. You can expand and collapse different blocks of code by using outlining. You can learn more about the code you are using by using IntelliSense, the **Object Browser**, and the Call Hierarchy. You can navigate inside your code by using features such as **Navigate To**, **Go To Definition**, and **Find All References**. You can insert blocks of code with code snippets, and you can generate code by using features such as **Generate From Usage**. If you have never used the Visual Studio 2015 editor before, see Editing Your Code for a quick overview.

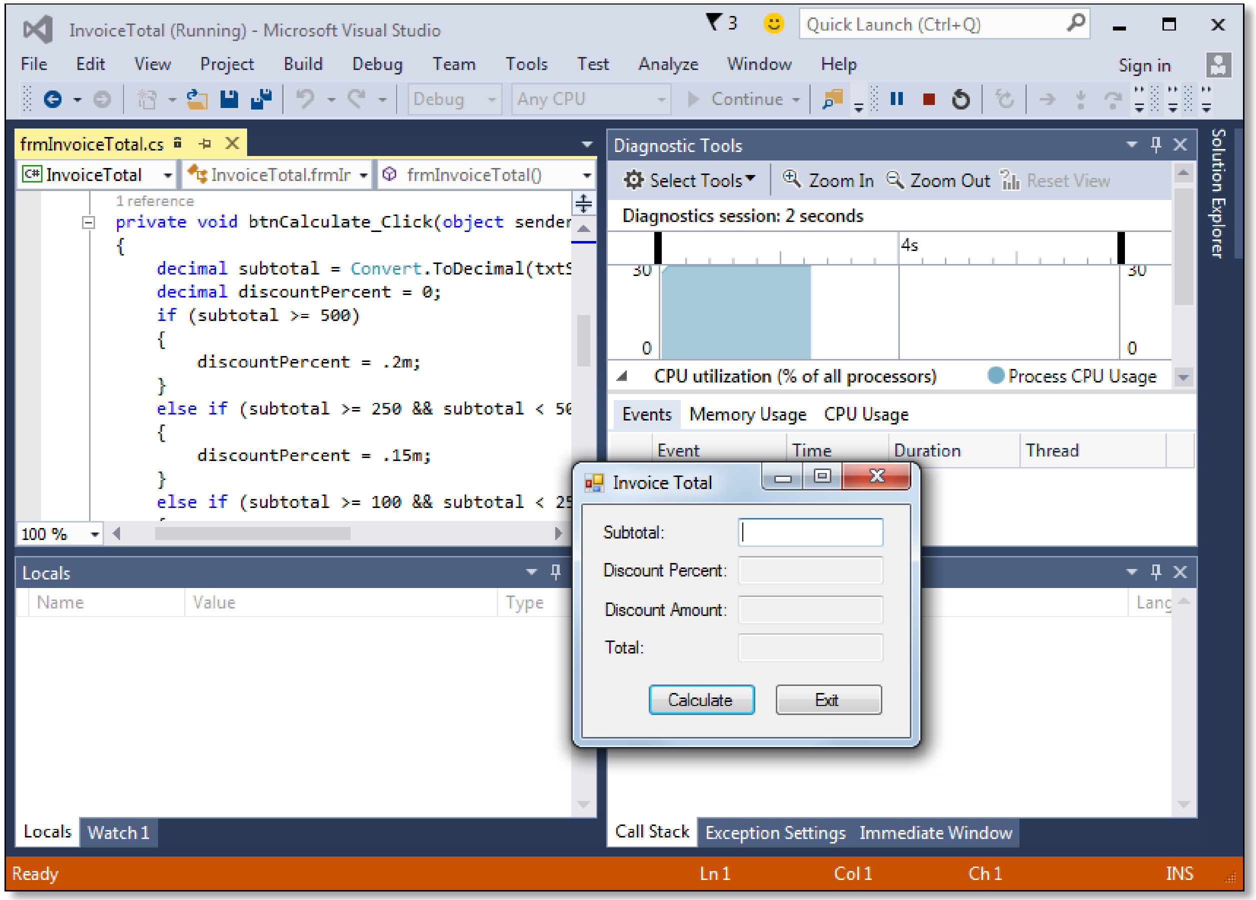
You can view your code in a number of different ways. To see a class view of your solution, you can open the **Class View** window or expand the nodes in the **Solution Explorer** under your class files.

You can search and replace text for single or multiple files. For more information, see Finding and Replacing Text. If you use regular expressions, note that find and replace now use .NET regular expressions. For more information, see Using Regular Expressions in Visual Studio.

Any Suggestions?
Print
Export (0)

IN THIS ARTICLE
- Editor Features
- Advanced Editing Features
- Navigating in the Code Window
- Customizing the Editor
- In this section
- See Also

The form that's displayed when you run the Invoice Total project


InvoiceTotal (Running) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Sign in
Debug
Any CPU
Continue
frmInvoiceTotal.cs
InvoiceTotal
InvoiceTotal.frmIr
frmInvoiceTotal()
Diagnostic Tools
Select Tools
Zoom In
Zoom Out
Reset View
Diagnostics session: 2 seconds
4s
30
0
CPU utilization (% of all processors)
Process CPU Usage
Events Memory Usage CPU Usage
Event Time Duration Thread
Solution Explorer
100 %
Locals
Name Value Type
Lang
Locals Watch 1
Call Stack Exception Settings Immediate Window
Ready Ln 1 Col 1 Ch 1 INS

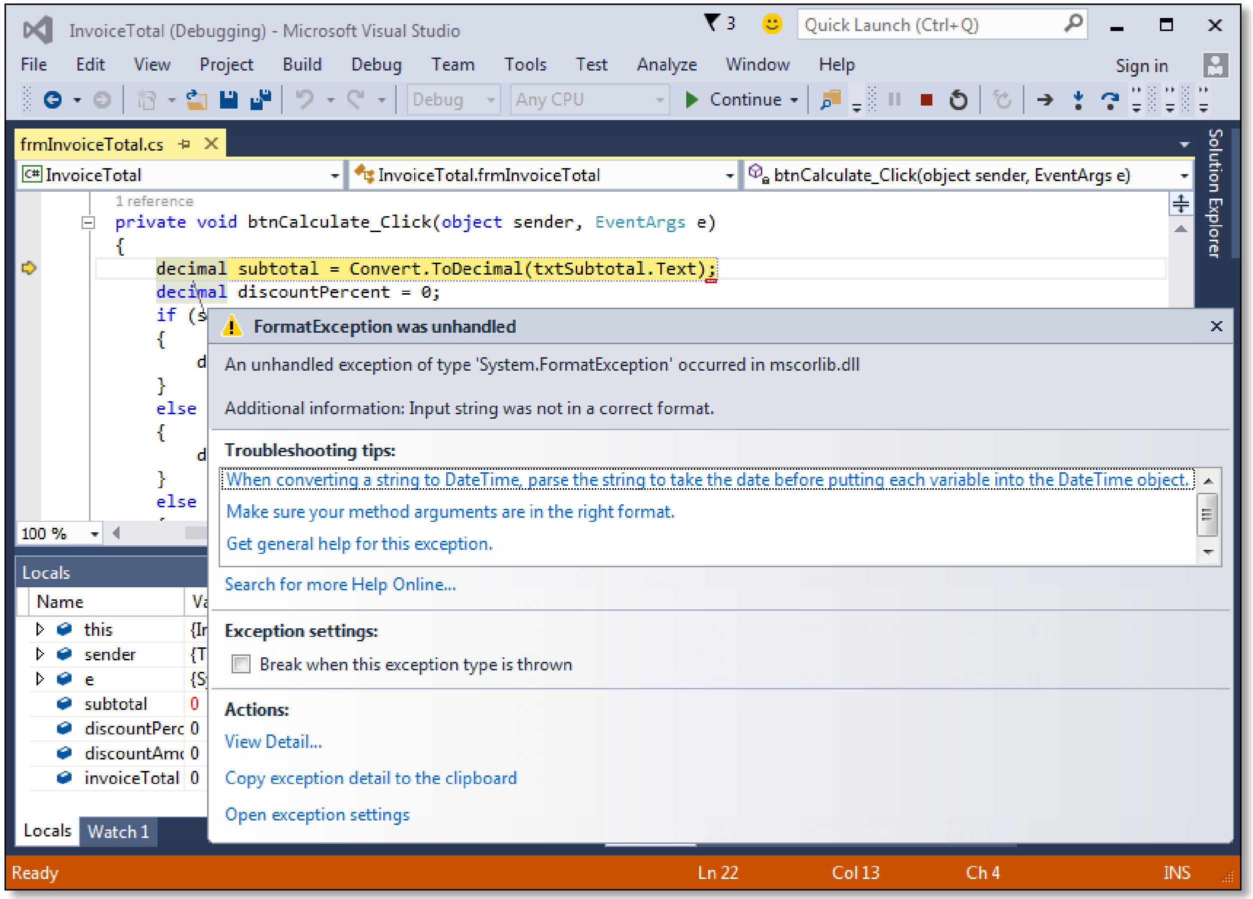
```
1 reference
private void btnCalculate_Click(object sender
{
    decimal subtotal = Convert.ToDecimal(txtS
    decimal discountPercent = 0;
    if (subtotal >= 500)
    {
        discountPercent = .2m;
    }
    else if (subtotal >= 250 && subtotal < 50
    {
        discountPercent = .15m;
    }
    else if (subtotal >= 100 && subtotal < 25
```

Invoice Total

Subtotal:	
Discount Percent:	
Discount Amount:	
Total:	

Calculate   Exit

The Exception Assistant that's displayed when a runtime error occurs



```
1 reference
private void btnCalculate_Click(object sender, EventArgs e)
{
    decimal subtotal = Convert.ToDecimal(txtSubtotal.Text);
    decimal discountPercent = 0;
    if (s
    {
        d
    }
    else
    {
        d
    }
    else
```

Name	Va
this	{Ir
sender	{T
e	{S
subtotal	0
discountPerc	0
discountAmc	0
invoiceTotal	0

FormatException was unhandled

An unhandled exception of type 'System.FormatException' occurred in mscorlib.dll

Additional information: Input string was not in a correct format.

Troubleshooting tips:

When converting a string to DateTime, parse the string to take the date before putting each variable into the DateTime object.

Make sure your method arguments are in the right format.

Get general help for this exception.

Search for more Help Online...

Exception settings:

☐ Break when this exception type is thrown

Actions:

View Detail...

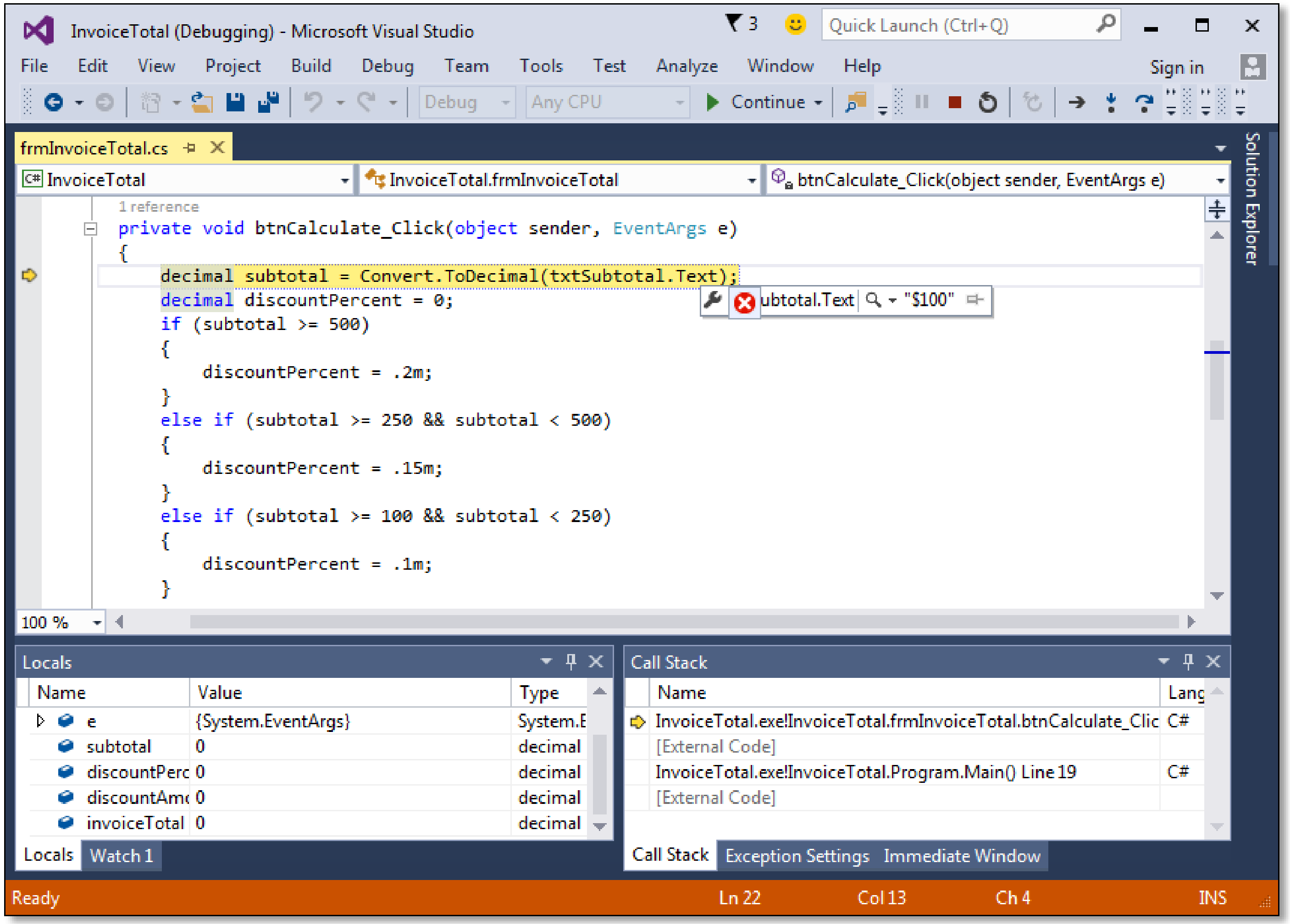
Copy exception detail to the clipboard

Open exception settings

How to test a project

1. Test the user interface, including the appearance of the controls, the tab order, the access keys, and the Enter and Esc keys.
2. Test valid input data.
3. Test invalid data or unexpected user actions.

How a project looks in break mode


InvoiceTotal (Debugging) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Sign in
Debug
Any CPU
Continue
frmInvoiceTotal.cs
InvoiceTotal
InvoiceTotal.frmInvoiceTotal
btnCalculate_Click(object sender, EventArgs e)
Solution Explorer
100 %

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

ubtotal.Text "$100"

Locals

Name	Value	Type
e	{System.EventArgs}	System.E
subtotal	0	decimal
discountPerc	0	decimal
discountAm	0	decimal
invoiceTotal	0	decimal

Locals Watch 1

Call Stack

Name	Lang
InvoiceTotal.exe!InvoiceTotal.frmInvoiceTotal.btnCalculate_Clic	C#
[External Code]	
InvoiceTotal.exe!InvoiceTotal.Program.Main() Line 19	C#
[External Code]	

Call Stack Exception Settings Immediate Window

Ready Ln 22 Col 13 Ch 4 INS