# C#
# C Sharp

# What is C#?

- C# is one of the programming languages supported by the .NET Framework's Common Language Runtime.

# What is .NET Framework ?

- The .NET Framework is a platform created by Microsoft for developing applications - Windows applications, Web applications, Web services, …

# The.NET Framework include the following:

- *The .NET languages:* These include Visual Basic, C#, JScript .NET (a server-side version of JavaScript), J# (a Java clone), and C++.

- *The Common Language Runtime (CLR):* This is the engine that executes all .NET programs and provides automatic services for these applications, such as security checking, memory management, and optimization.

- *The .NET Framework class library:* The class library collects thousands of pieces of prebuilt functionality that you can “snap in” to your applications. These features are sometimes organized into technology sets, such as ADO.NET (the technology for creating database applications) and Windows Forms (the technology for creating desktop user interfaces).

- *ASP.NET:* This is the engine that hosts the web applications you create with .NET, and supports almost any feature from the .NET class library. ASP.NET also includes a set of web-specific services, like secure authentication and data storage.

- *Visual Studio:* This optional development tool contains a rich set of productivity and debugging features. The Visual Studio setup DVD includes the complete .NET Framework, so you won’t need to download it separately.

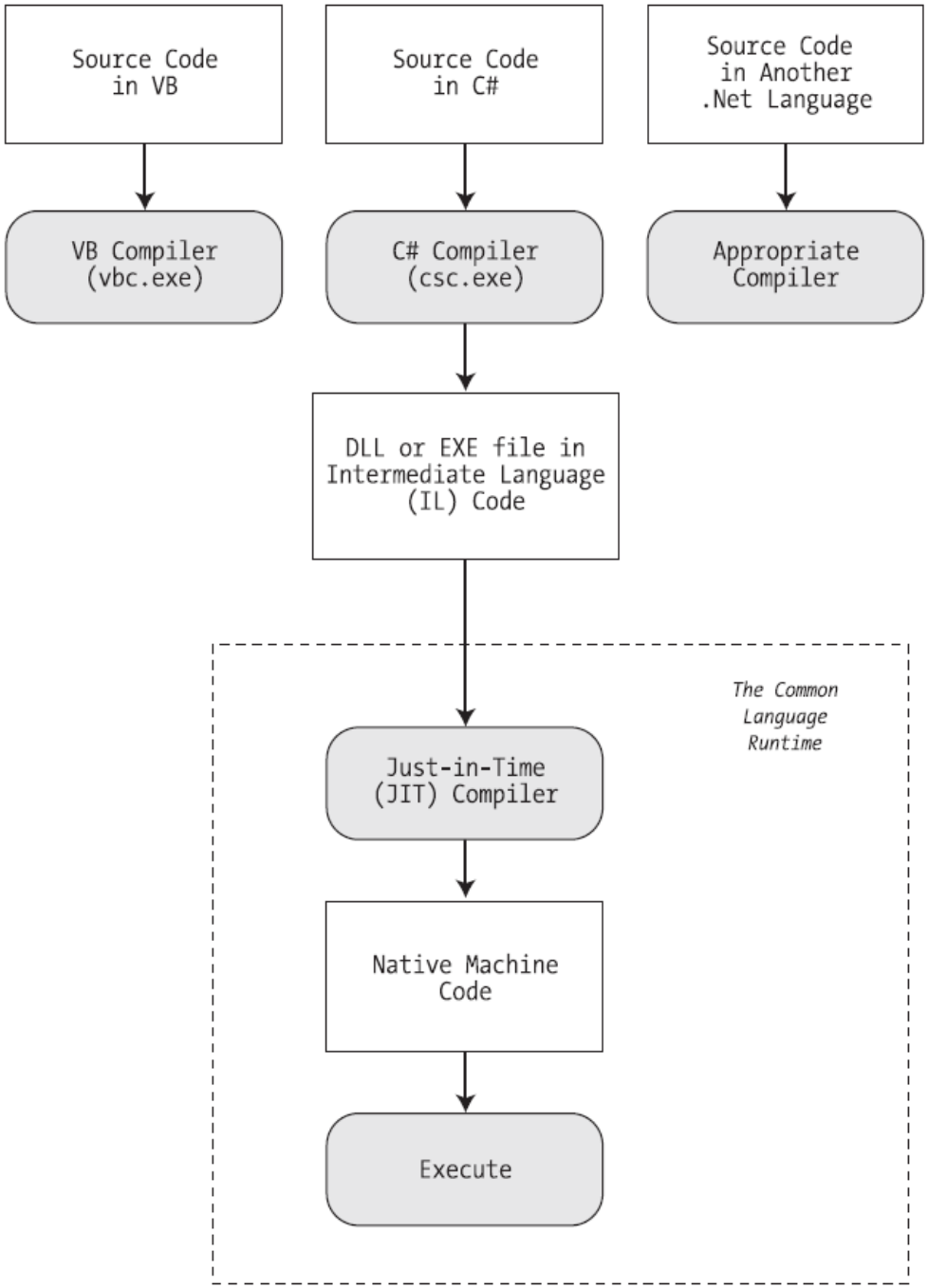
# What is CIL?

- Before the code is executed, all the .NET languages are compiled into another lower-level language - Common Intermediate Language (CIL).

- CIL code isn’t specific to any operating system.

# What is JIT?

- *Just-in-Time (JIT)* compiler compiles CIL into native code that is specific to the OS and machine architecture being targeted. Only at this point can the OS execute the application. The *just-in-time* part of the name reflects the fact that CIL code is only compiled as, and when, it is needed.


Source Code in VB
Source Code in C#
Source Code in Another .Net Language
VB Compiler (vbc.exe)
C# Compiler (csc.exe)
Appropriate Compiler
DLL or EXE file in Intermediate Language (IL) Code
The Common Language Runtime
Just-in-Time (JIT) Compiler
Native Machine Code
Execute

# Beginning C#

1. **On the *File menu, point to New, and then click Project.***

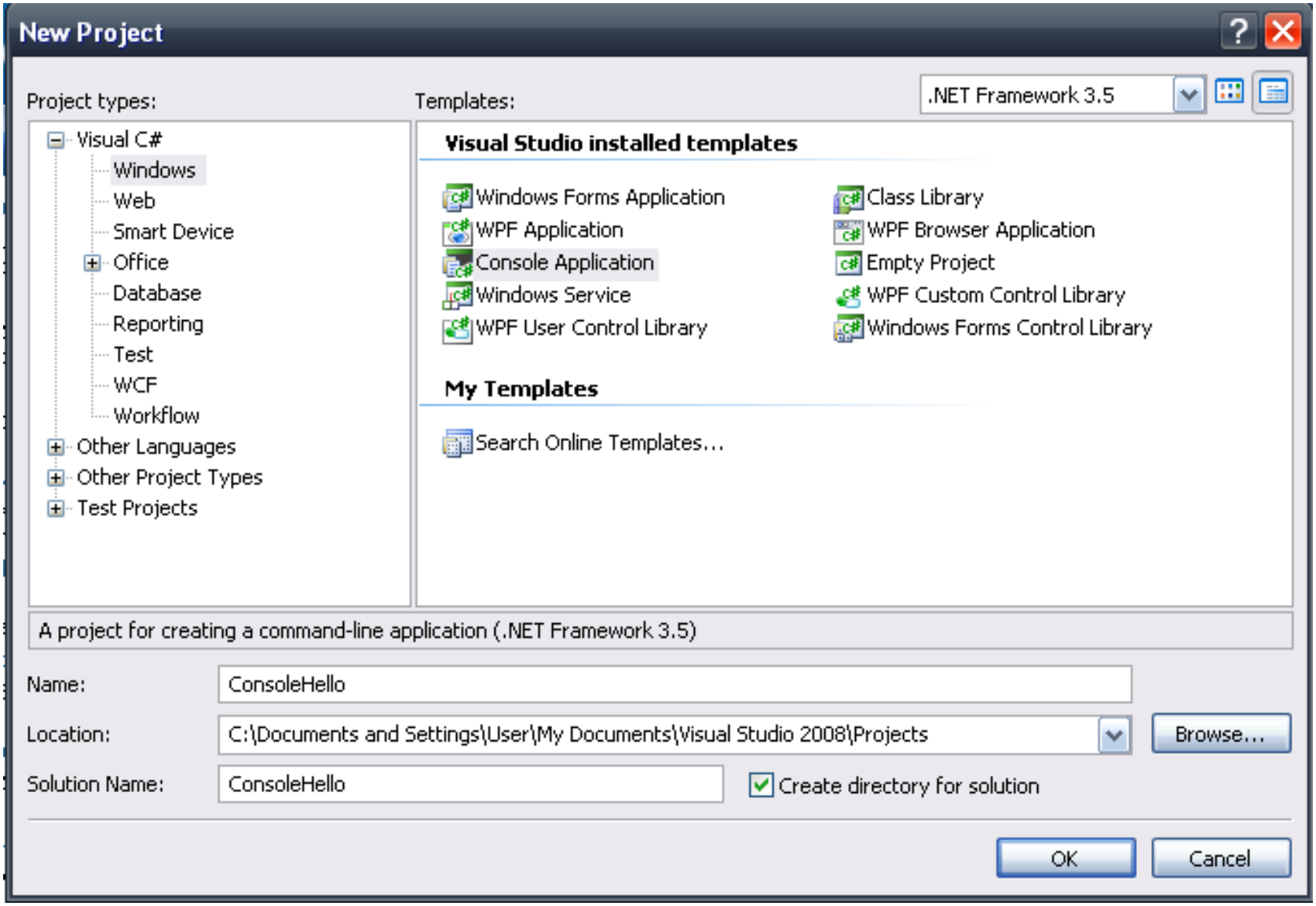
   The *New Project* dialog box opens. This dialog box lists the templates that you can use as a starting point for building an application. The dialog box categorizes templates according to the programming language you are using and the type of application.

**2.** In the *Project types* pane, click *Visual C#*. In the *Templates pane*, click the *Console Application* icon.

**3.** In the *Location field* specify the location of your project files.

**4.** In the *Name field,* type ConsoleHello*.*

**5.** Ensure that the Create directory for solution check box is selected, and then click OK.


New Project
Project types:
Templates:
.NET Framework 3.5
Visual C#
Windows
Web
Smart Device
Office
Database
Reporting
Test
WCF
Workflow
Other Languages
Other Project Types
Test Projects
Visual Studio installed templates
Windows Forms Application
WPF Application
Console Application
Windows Service
WPF User Control Library
Class Library
WPF Browser Application
Empty Project
WPF Custom Control Library
Windows Forms Control Library
My Templates
Search Online Templates...
A project for creating a command-line application (.NET Framework 3.5)
Name:
ConsoleHello
Location:
C:\Documents and Settings\User\My Documents\Visual Studio 2008\Projects
Browse...
Solution Name:
ConsoleHello
Create directory for solution
OK
Cancel

# Solution explorer

- Solution ‘ConsoleHello’ This is the top-level solution file, of which there is one per application. If you use Windows Explorer to look at the location of your project files folder, you’ll see that the actual name of this file is ConsoleHello.sln. Each solution file contains references to one or more project files.

- ConsoleHello is the C# project file. Each project file references one or more files containing the source code and other items for the project. All the source code in a single project must be written in the same programming language.

- In Windows Explorer, this file is actually called ConsoleHello.csproj.

- Properties is a folder in the ConsoleHello project. It contains a file called AssemblyInfo.cs that you can use to add attributes to a program, such as the name of the author, the date the program was written, ...

- References is a folder that contains references to compiled code that your application can use.

- When code is compiled, it is converted into an assembly and given a unique name.

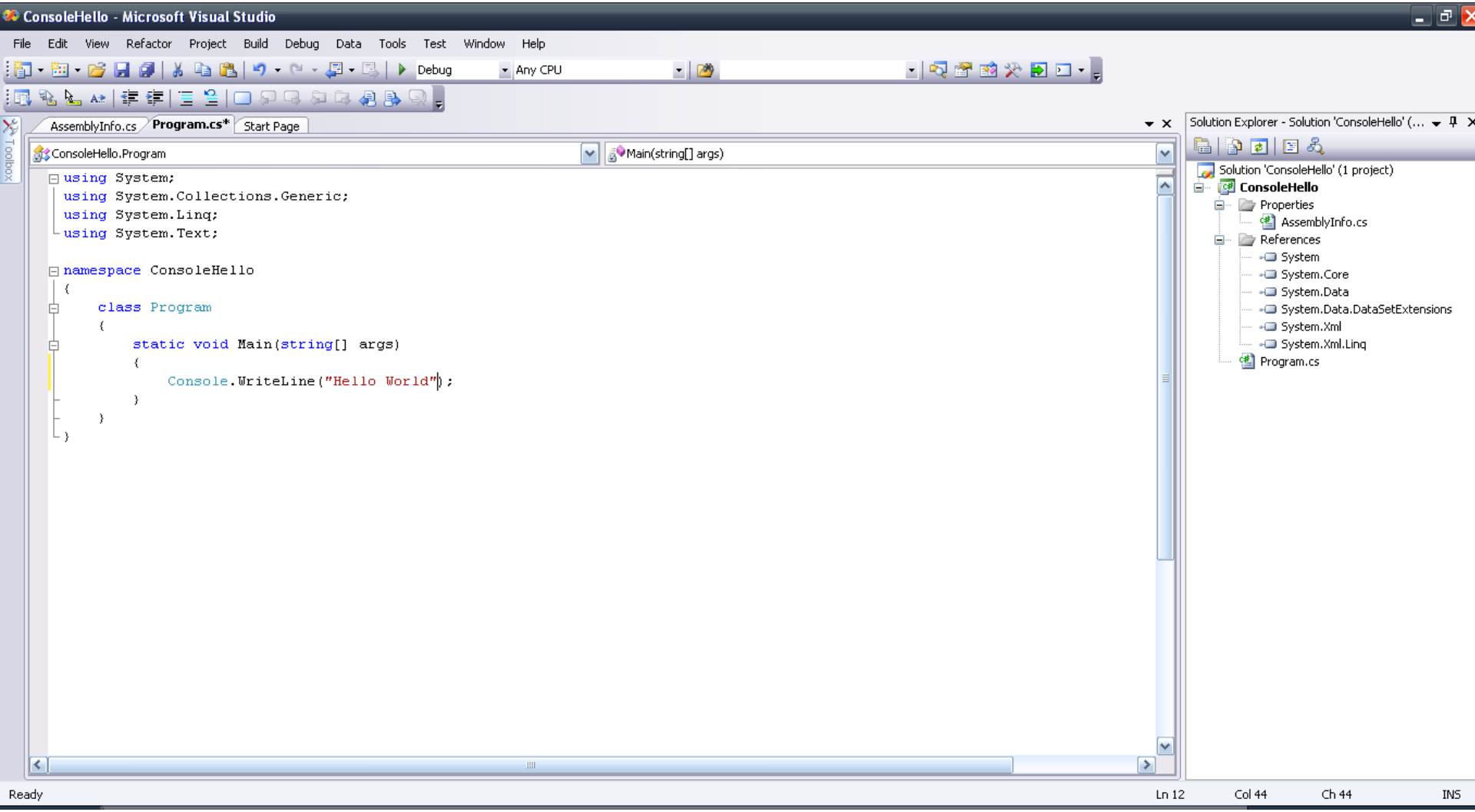
- Developers use assemblies to package useful bits of code they have written so they can distribute it to other developers who might want to use the code in their applications.

- Program.cs is a C# source file and is the one currently displayed in the Code and Text Editor window when the project is first created.

- You will write your code for the console application in this file.

- The Program.cs file defines a class called Program that contains a method called Main. All methods must be defined inside a class.

- The *Main* method is special—it designates the program's entry point. It must be a static method.


ConsoleHello - Microsoft Visual Studio
File Edit View Refactor Project Build Debug Data Tools Test Window Help
Debug
Any CPU
AssemblyInfo.cs
Program.cs*
Start Page
ConsoleHello.Program
Main(string[] args)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleHello
{
class Program
{
static void Main(string[] args)
{
Console.WriteLine("Hello World");
}
}
}
Solution Explorer - Solution 'ConsoleHello' (...
Solution 'ConsoleHello' (1 project)
ConsoleHello
Properties
AssemblyInfo.cs
References
System
System.Core
System.Data
System.Data.DataSetExtensions
System.Xml
System.Xml.Linq
Program.cs
Ready
Ln 12
Col 44
Ch 44
INS

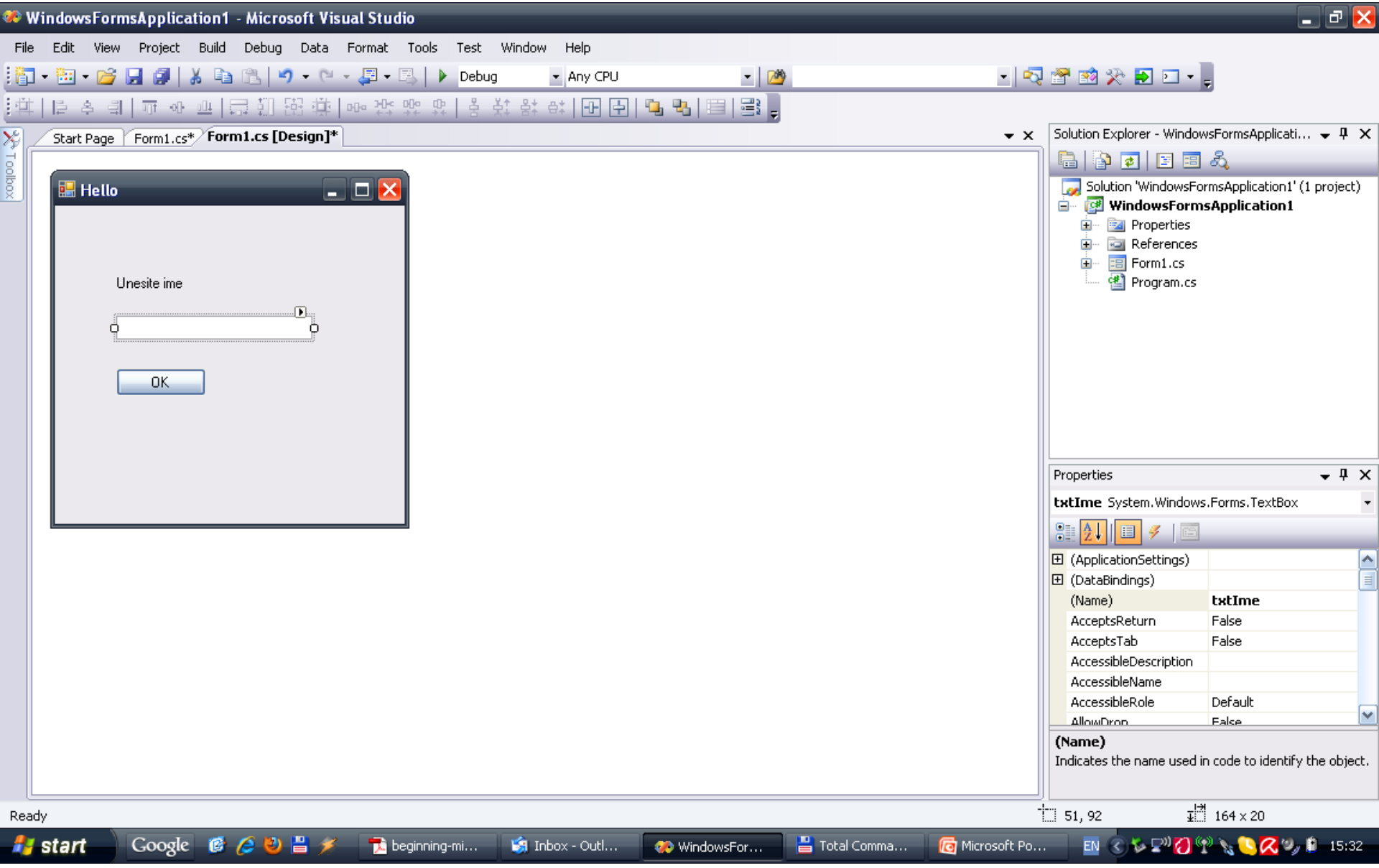
- *Consolе* is a built-in class that contains the methods for displaying messages on the screen and getting input from the keyboard.

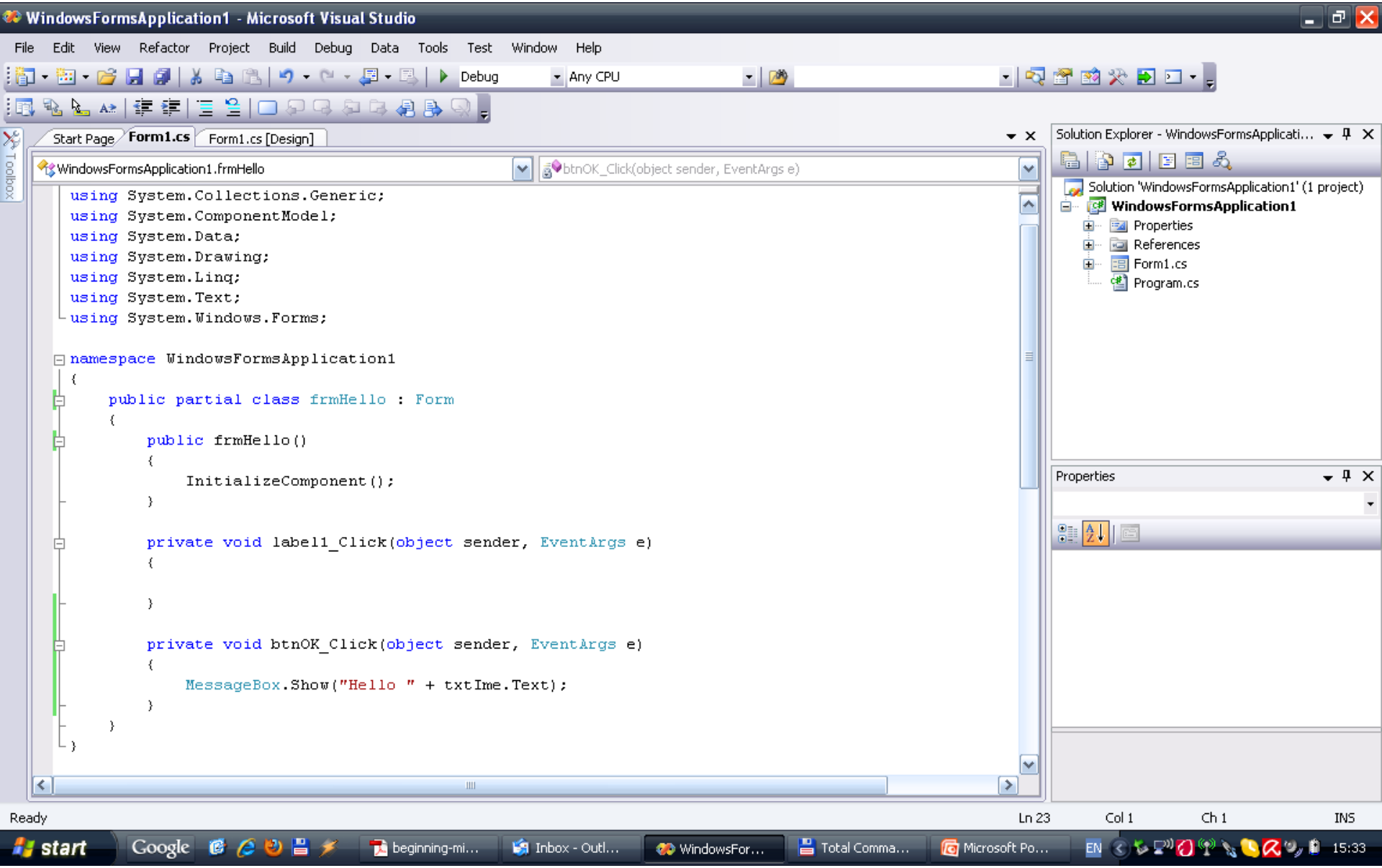
# Namespaces

- Named container for other identifiers, such as classes.
- Two classes with the same name will not be confused with each other if they live in different namespaces.
- Visual Studio 2008 environment is using the name of your project as the top-level namespace.
- A *using* statement brings a namespace into scope.

```
namespace LevelOne
{
        using LevelTwo;
        namespace LevelTwo
        {
            // name “NameTwo” defined
        }
}
```

Code in the LevelOne namespace can now refer to LevelTwo.NameTwo by simply using NameTwo.


WindowsFormsApplication1 - Microsoft Visual Studio
File Edit View Project Build Debug Data Format Tools Test Window Help
Debug
Any CPU
Start Page
Form1.cs*
Form1.cs [Design]*
Toolbox
Hello
Unesite ime
OK
Solution Explorer - WindowsFormsApplicati...
Solution 'WindowsFormsApplication1' (1 project)
WindowsFormsApplication1
Properties
References
Form1.cs
Program.cs
Properties
txtIme System.Windows.Forms.TextBox
(ApplicationSettings)
(DataBindings)
(Name) txtIme
AcceptsReturn False
AcceptsTab False
AccessibleDescription
AccessibleName
AccessibleRole Default
AllowDrop False
(Name)
Indicates the name used in code to identify the object.
Ready
51, 92
164 x 20
start
Google
beginning-mi...
Inbox - Outl...
WindowsFor...
Total Comma...
Microsoft Po...
EN
15:32


WindowsFormsApplication1 - Microsoft Visual Studio
File Edit View Refactor Project Build Debug Data Tools Test Window Help
Debug
Any CPU
Start Page
Form1.cs
Form1.cs [Design]
WindowsFormsApplication1.frmHello
btnOK_Click(object sender, EventArgs e)
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
public partial class frmHello : Form
{
public frmHello()
{
InitializeComponent();
}
private void label1_Click(object sender, EventArgs e)
{
}
private void btnOK_Click(object sender, EventArgs e)
{
MessageBox.Show("Hello " + txtIme.Text);
}
}
}
Solution Explorer - WindowsFormsApplicati...
Solution 'WindowsFormsApplication1' (1 project)
WindowsFormsApplication1
Properties
References
Form1.cs
Program.cs
Properties
Ready
Ln 23
Col 1
Ch 1
INS
start
Google
beginning-mi...
Inbox - Outl...
WindowsFor...
Total Comma...
Microsoft Po...
EN
15:33


Hello
Unesite ime
James Bond
OK
Hello James Bond
OK

# Variables

- You can think of variables in computer memory as boxes sitting on a shelf. You can put things (data) in boxes (variables) and take them out again.

# ***Variable Naming***

- The first character of a variable name must be either a letter, an underscore character (_), or the *at symbol ( @ ).*

- Subsequent characters may be letters, underscore characters, or numbers.

- For example, the following variable names are fine:

```
myBigVar
VAR1
_test
```

- These are not, however:

```
99BottlesOfBeer
namespace
It’s-All-Over
```

# *Naming Conventions*

- Currently, two naming conventions are used in the .NET Framework namespaces:

- *PascalCase*
- *camelCase .*

- The case used in the names indicates their usage. They both apply to names that are made up of multiple words and they both specify that each word in a name should be in lowercase except for its first letter, which should be uppercase. For camelCase terms, there is an additional rule: The first word should start with a lowercase letter.

The following are camelCase variable names:

- `age`
- `firstName`
- `timeOfDeath`

These are PascalCase:

- `Age`
- `LastName`
- `WinterOfDiscontent`

# Types

- Variables come in different types.

- The reasoning behind this type system is that different types of data may require different methods of manipulation, and by restricting variables to individual types you can avoid mixing them up.

| Type | Alias For | Allowed Values |
|---|---|---|
| `sbyte` | `System.SByte` | Integer between –128 and 127 |
| `byte` | `System.Byte` | Integer between 0 and 255 |
| `short` | `System.Int16` | Integer between –32768 and 32767 |
| `ushort` | `System.UInt16` | Integer between 0 and 65535 |
| `int` | `System.Int32` | Integer between –2147483648 and 2147483647 |
| `uint` | `System.UInt32` | Integer between 0 and 4294967295 |
| `long` | `System.Int64` | Integer between –9223372036854775808 and 9223372036854775807 |
| `ulong` | `System.UInt64` | Integer between 0 and 18446744073709551615 |

| Type | Alias For | Min M | Max M | Min E | Max E | Approx. Min Value | Approx. Max Value |
|---|---|---|---|---|---|---|---|
| `float` | `System.Single` | 0 | 224 | −149 | 104 | $1.5\times10^{-45}$ | $3.4\times10^{38}$ |
| `double` | `System.Double` | 0 | 253 | −1075 | 970 | $5.0\times10^{-324}$ | $1.7\times10^{308}$ |
| `decimal` | `System.Decimal` | 0 | 296 | −26 | 0 | $1.0\times10^{-28}$ | $7.9\times10^{28}$ |

| Type | Alias For | Allowed Values |
|---|---|---|
| `char` | `System.Char` | Single Unicode character, stored as an integer between 0 and 65535 |
| `bool` | `System.Boolean` | Boolean value, `true` or `false` |
| `string` | `System.String` | A sequence of characters |

# *Variable Declaration and Assignment*

- To use variables, you have to *declare* them. This means that you have to assign them a name and a type. Once you have declared variables, you can use them as storage units for the type of data that you declared them to hold.
- C# syntax for declaring variables

```
< type > < name > ;
```

- Declared variables must be initialized before you use them.

- Declaration: `int age;`
- Initialization: `age = 22;`

OR

- Declaration and initialization:

```
int age  = 22;
```

# *Mathematical Operators*

| Operator | Category | Example Expression | Result |
|---|---|---|---|
| + | Binary | `var1 = var2 + var3;` | var1 is assigned the value that is the sum of var2 and var3. |
| - | Binary | `var1 = var2 - var3;` | var1 is assigned the value that is the value of var3 subtracted from the value of var2. |
| * | Binary | `var1 = var2 * var3;` | var1 is assigned the value that is the product of var2 and var3. |
| / | Binary | `var1 = var2 / var3;` | var1 is assigned the value that is the result of dividing var2 by var3. |
| % | Binary | `var1 = var2 % var3;` | var1 is assigned the value that is the remainder when var2 is divided by var3. |
| + | Unary | `var1 = +var2;` | var1 is assigned the value of var2. |
| - | Unary | `var1 = -var2;` | var1 is assigned the value of var2 multiplied by -1. |

| Operator | Category | Example Expression | Result |
|---|---|---|---|
| ++ | Unary | `var1 = ++var2;` | var1 is assigned the value of var2 + 1. var2 is incremented by 1. |
| -- | Unary | `var1 = --var2;` | var1 is assigned the value of var2 - 1. var2 is decremented by 1. |
| ++ | Unary | `var1 = var2++;` | var1 is assigned the value of var2. var2 is incremented by 1. |
| -- | Unary | `var1 = var2--;` | var1 is assigned the value of var2. var2 is decremented by 1. |

# Exception:

- The binary + operator does make sense when used with string type variables.
- None of the other mathematical operators, however, work with strings.

| Operator | Category | Example Expression | Result |
|---|---|---|---|
| + | Binary | `var1 = var2 + var3;` | `var1` is assigned the value that is the concatenation of the two strings stored in `var2` and `var3`. |

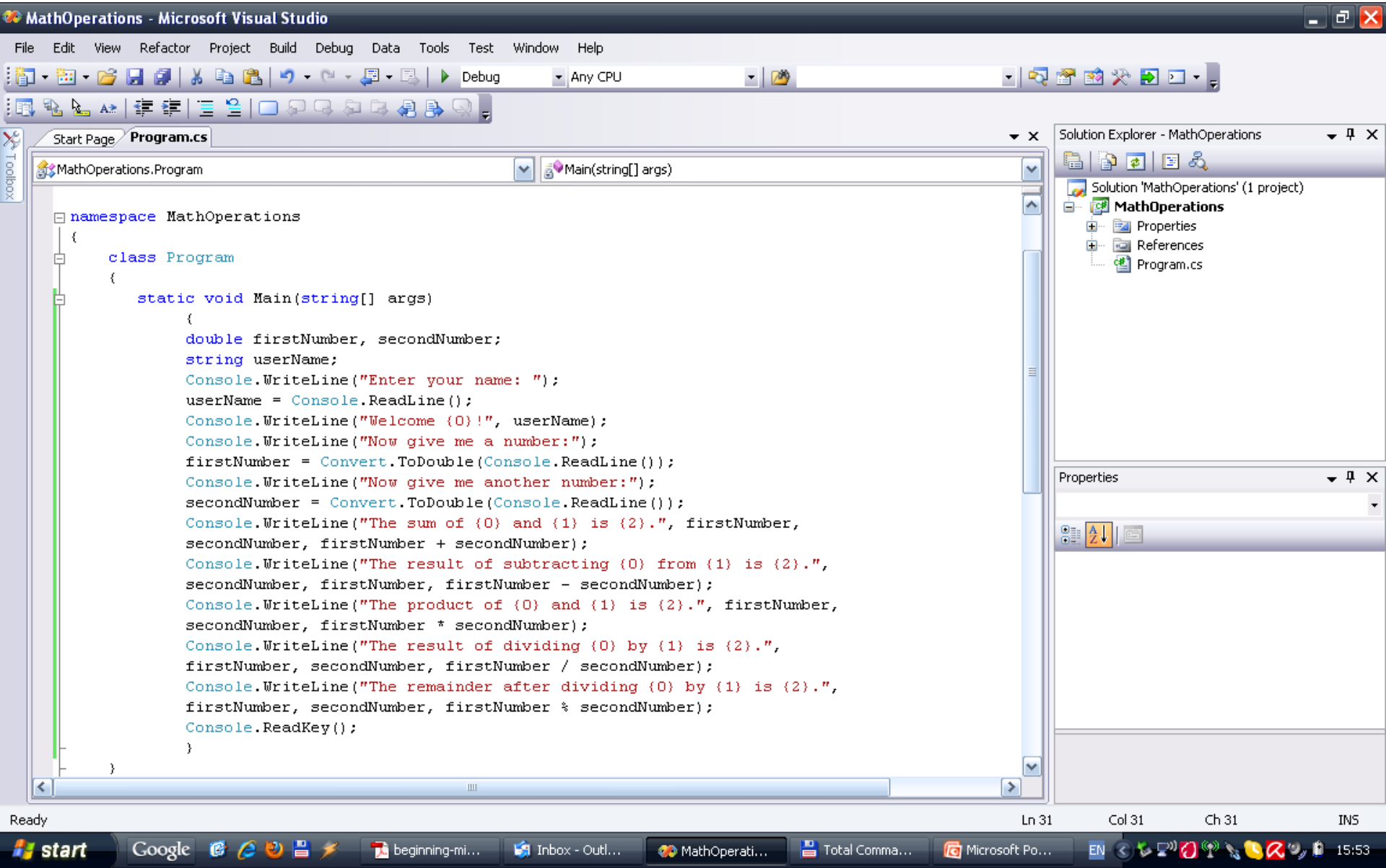
# *Assignment Operators*

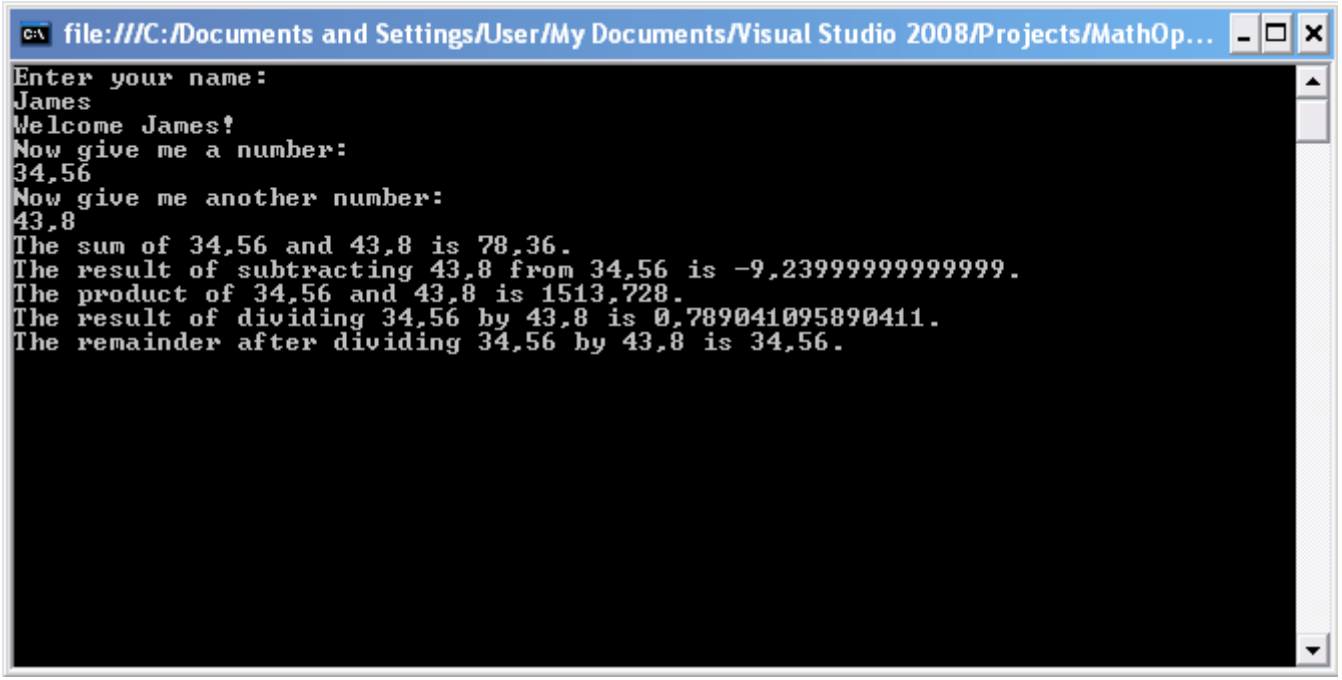
| Operator | Category | Example Expression | Result |
|---|---|---|---|
| = | Binary | `var1 = var2;` | var1 is assigned the value of var2. |
| += | Binary | `var1 += var2;` | var1 is assigned the value that is the sum of var1 and var2. |
| -= | Binary | `var1 -= var2;` | var1 is assigned the value that is the value of var2 subtracted from the value of var1. |
| *= | Binary | `var1 *= var2;` | var1 is assigned the value that is the product of var1 and var2. |
| /= | Binary | `var1 /= var2;` | var1 is assigned the value that is the result of dividing var1 by var2. |
| %= | Binary | `var1 %= var2;` | var1 is assigned the value that is the remainder when var1 is divided by var2. |

# *Operator Precedence*

| Precedence | Operators |
|---|---|
| Highest | ++, -- (used as prefixes); +, - (unary) |
| | *, /, % |
| | +, - |
| | =, *=, /=, %=, +=, -= |
| Lowest | ++, -- (used as suffixes) |

# Primer


MathOperations - Microsoft Visual Studio
File Edit View Refactor Project Build Debug Data Tools Test Window Help
Debug
Any CPU
Start Page
Program.cs
MathOperations.Program
Main(string[] args)
namespace MathOperations
{
class Program
{
static void Main(string[] args)
{
double firstNumber, secondNumber;
string userName;
Console.WriteLine("Enter your name: ");
userName = Console.ReadLine();
Console.WriteLine("Welcome {0}!", userName);
Console.WriteLine("Now give me a number:");
firstNumber = Convert.ToDouble(Console.ReadLine());
Console.WriteLine("Now give me another number:");
secondNumber = Convert.ToDouble(Console.ReadLine());
Console.WriteLine("The sum of {0} and {1} is {2}.", firstNumber,
secondNumber, firstNumber + secondNumber);
Console.WriteLine("The result of subtracting {0} from {1} is {2}.",
secondNumber, firstNumber, firstNumber - secondNumber);
Console.WriteLine("The product of {0} and {1} is {2}.", firstNumber,
secondNumber, firstNumber * secondNumber);
Console.WriteLine("The result of dividing {0} by {1} is {2}.",
firstNumber, secondNumber, firstNumber / secondNumber);
Console.WriteLine("The remainder after dividing {0} by {1} is {2}.",
firstNumber, secondNumber, firstNumber % secondNumber);
Console.ReadKey();
}
}
Solution Explorer - MathOperations
Solution 'MathOperations' (1 project)
MathOperations
Properties
References
Program.cs
Properties
Ready
Ln 31
Col 31
Ch 31
INS
start
Google
beginning-mi...
Inbox - Outl...
MathOperati...
Total Comma...
Microsoft Po...
EN
15:53


file:///C:/Documents and Settings/User/My Documents/Visual Studio 2008/Projects/MathOp...
Enter your name:
James
Welcome James!
Now give me a number:
34,56
Now give me another number:
43,8
The sum of 34,56 and 43,8 is 78,36.
The result of subtracting 43,8 from 34,56 is -9,23999999999999.
The product of 34,56 and 43,8 is 1513,728.
The result of dividing 34,56 by 43,8 is 0,7890410958904111.
The remainder after dividing 34,56 by 43,8 is 34,56.

# Explanation

- For user input use `Console.ReadLine()` . This command prompts the user for input, which is stored in a string variable.

```
userName = Console.ReadLine();
```

- Use the command `Convert.ToDouble()` on a string obtained by `Console.ReadLine()` to convert the string into a double type.

```
firstNumber = Convert.ToDouble(Console.ReadLine());
```

```
Console.WriteLine(“The sum of {0} and {1} is {2}.”, firstNumber,
secondNumber, firstNumber + secondNumber);
```

- The integers start at 0 and are incremented by 1, and the total number of placeholders should match the number of variables specified in the comma - separated list following the string. When the text is output to the console, each placeholder is replaced by the corresponding value for each variable. In the preceding example, the {0} is replaced with the actual value of the first variable, `firstNumber` , {1} is replaced with the contents of `secondNumber` , and {2} with result of operation `firstNumber + secondNumber`