Instructor: Bruce Reynolds

Introduction to Windows and Web Applications in C# Class 1

Class 1

- Class Overview
- The .NET Framework
- Hello, World
- Programming Concepts

Section 1: Class Overview

- Language – C# language elements and programming concepts, including types, variables, garbage collection, expressions, statements, exceptions, classes, structures, and enumerations.
- .NET Framework – The Common Language Runtime and the .NET Framework Class Library

Class Overview

- Windows - Event-driven Windows Forms applications, and common controls such as the TextBox, ListBox, Label, Checkbox, and ToolStrip.
- Data - Retrieval and display/update of date from SQL Server.
- Web – Event-driven web pages.

Materials

- Required
  - Microsoft Visual Studio Professional, 2010.
    - Available for free download from Dreamspark
    - Included with your enrollment
  - SQL Server 2008
    - An express version is automatically installed with VS
  - C# 2010 for Programmers 4/e, Deitel & Deitel
  - Internet and email access
  - Ability to view PDF files

Language Specification

- The language spec is available on MSDN
  - http://msdn.microsoft.com/en-us/library/ms228593.aspx
  - Can be downloaded to your own computer
  - Also available in your Visual Studio installation directory as a Word document
    - If you do not have Word installed, download a free Word viewer from Microsoft

Other Books

- The C# Programming Language 4/e. Hejlsberg, Torgersen, Wiltamuth, Golde. Addison Wesley, 2010.
- Programming C# 4.0. Griffiths, Adams, Liberty. O'Reilly, 2010.
- Pro C# 5.0 and the .NET 4.5 Framework. Troelsen. Apress, 2012.

How to Contact Me

- Bruce's email: uwinstructor@frontier.com
- Course Moodle website: http://moodle.extn.washington.edu/course/view.php?id=3535

Grading

- There will be eight assignments
  - Some assignments can be completed in class
- To receive a grade of "SC" Successful Completion in the course:
  - Attend at least eight classes
  - Turn in all eight assignments
    - Or get your assignment checked off in class

Syllabus

Class	Topic
1	Visual Studio, Programming Concepts
2	.NET Framework, Types, Control Structures
3, 4 & 5	Classes, Arrays and Collections, Files, Exceptions
6 & 7	Windows applications, Controls
8 & 9	Databases, Data sources
10	Web Programming, Office Automation

Section 2: The .NET Framework

What is a Computer

- A computer consists of hardware and the software that controls the hardware.
  - Hardware: mouse, display, memory, CPU, disk
  - Software:
    - The Operating System controls the execution of applications and their access to the hardware.
    - Applications: Task-specific programs

Software

- Applications – Task-specific programs
  - CLR – Common language runtime
  - .NET Framework
  - Visual Studio
  - Managed applications (C#)

What is .NET and the CLR?

- The CLR provides a platform with:
  - Type-safety
  - Garbage collection
  - Language independence
  - Metadata
  - Common intermediate language
  - Assemblies

What is the .NET Framework?

- .NET Framework class library
  - An API for writing programs that run on the CLR
  - Organized into namespaces
  - Designed to be used from any .NET-aware programming language, such as C# and VB
    - Many other languages are supported, including C++, Java, Fortran, Perl, Python, COBOL, Eiffel, etc.

Programming Languages

- A programming language is an artificial language that can be used to write programs which control the behavior of a machine, particularly a computer. (Wikipedia)
  - Machine language – the language understood by the CPU, typically numbers
  - Assembly language – an English-like language that's translated to machine language
  - Higher level languages – languages that provide a higher level of abstraction for specific programming tasks. **(C# - a general programming language)**

The C# Language

- "C# (pronounced "See Sharp") is a simple, modern, object-oriented, and type-safe programming language." (From the C# specification.)
- **Simple, modern** – based on the C language.
- **Object-oriented**
  - Objects are things in the problem domain.
  - An object-oriented language supports development of applications based on the objects in the problem.
- **Type-safe** – reduces programming errors

Visual Studio

- A tool for creating C# applications. (You could use notepad and the command line compiler.)
- Application types
  - Console applications
  - Windows Forms
  - WPF – Windows Presentation Foundation
  - Web applications
  - Devices

Section 3: Hello, World

DEMO: Hello, World

- Open Visual Studio, selecting the C# settings.
- File, New, Project
- Windows, Console Application, OK
- Ctrl+F5

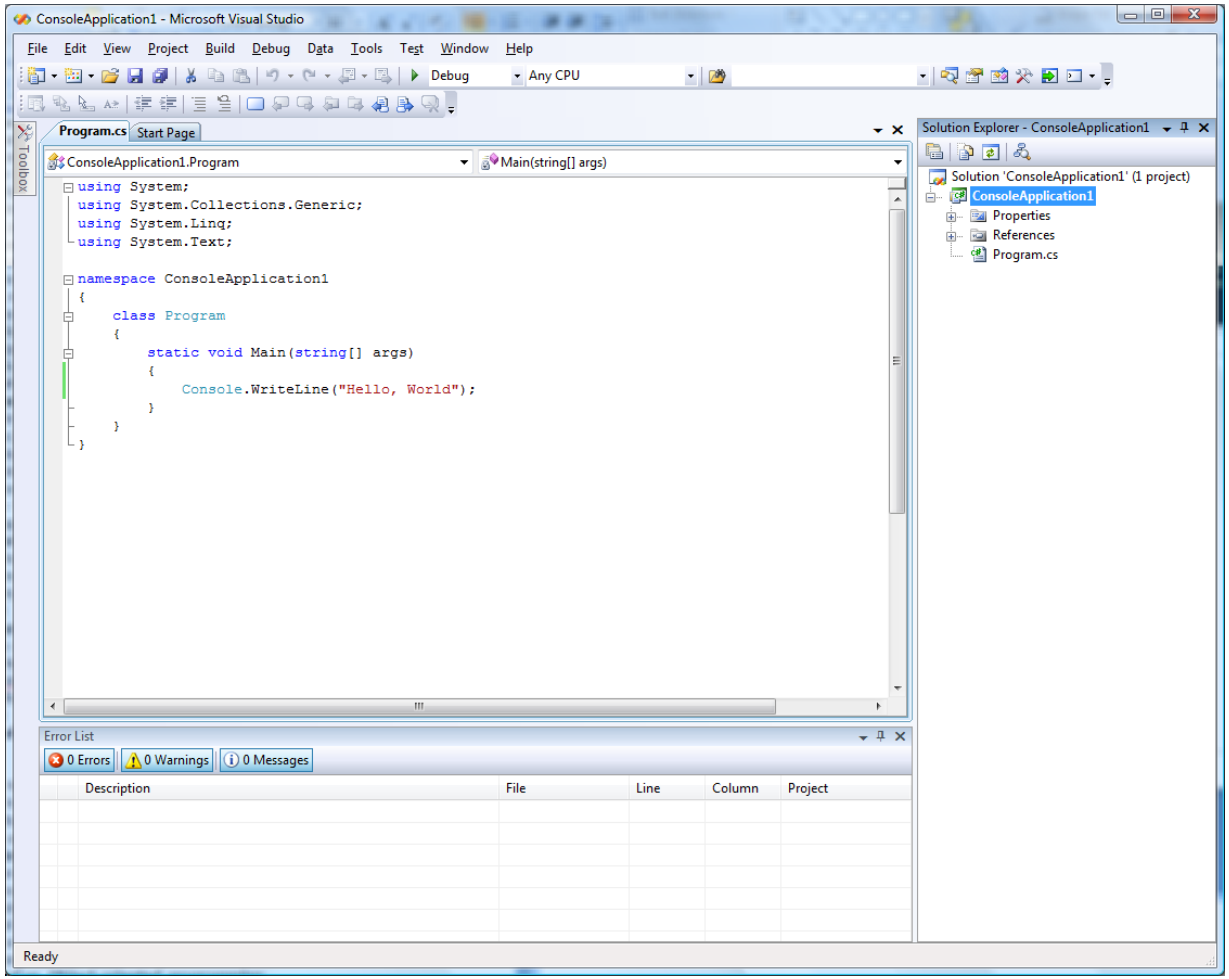
Hello, World

- Add this code inside the {} (curly braces) under `static void Main`:

```
// Write Hello, World
Console.WriteLine("Hello, World");
```

- Notice the straight quotes
- Ctrl+F5

The Visual Studio IDE



IDE

- Menu options – depend on your settings
- Solution Explorer
- Code editor
- Help system
- Error list
- Toolbox

Compiling

- With debugging – F5
  - Add a breakpoint by clicking in the margin
- Without debugging – Ctrl+F5

The Hello World Code

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            // Write Hello, World
            Console.WriteLine("Hello, World");
        }
    }
}
```

Console applications

- Console applications are text-based applications with no graphical user interface (GUI). They are also called command line applications.
- Input and output are handled through the keyboard and a console window.
- We'll use console apps for the first several weeks to separate language concepts from technology concepts.

Comments

- Comments are lines of code that are not compiled and executed.
- There are two styles of C# comments:
  - //
  - /* */
- C# also supports documentation comments that can be converted into documentation files.

The Console class

- A class represents an object in the problem. In this case, the Console class (also called the Console object) represents the console window.
- Classes are the building blocks of object-oriented programming and the .NET Framework.

The WriteLine method

- Methods define what a class/object can do.
  - The .NET Framework classes have methods.
  - You will write code that is contained in methods.
- Parameters, separated by commas, are enclosed in the parentheses following the method name.
  - The parameter in this case is a string, “Hello, World”, shown in red.

Fun Things to Do with Console

- Experiment with IntelliSense and F1.
- Console.ForegroundColor and ConsoleColor - change the text color with Reset the colors
- Console.Write - write without the carriage return.
- Console.Beep
- Console.SetWindowSize - set the window size.

C# code

- The semicolon (;)
  - The semicolon marks the end of statement in C#.
- The curly braces {}
  - Curly braces define a group of statements that are executed as a group.
- The dot operator (.)
  - The dot operator is used to access the methods (and properties) of a class.
- Case sensitivity
  - C# is case sensitive. “Console” and “console” are different.

static void Main

- "static void Main" is a method in the class Program.
  - It is the entry point for the application.
  - "Main" is the name of the application.
  - We'll cover "static" and "void" in later lectures. These are required for the entry point method.

The Program Class

- The Hello, World application contains a class that you wrote, named “Program”.
- All code in a C# application must be contained in the members of a class.
  - The Program class contains one member, “Main”. Other types of members are properties, constructors, and events.

The ConsoleApplication1 Namespace

- Namespaces are labels used to organize classes and other types.

Using

- The using keyword lets you write Console instead of System.Console.
- Check the help to find the namespace for the Console class.

Code Editor Tour

- IntelliSense
- Fonts and colors
- Collapse and expand
- Comment and uncomment
- Pretty listing
- Snippets
- Compiler errors - squigglies

Section 4: Programming Concepts

Program Flow

- There are three basic concepts for program flow:
  - Sequence
  - Selection
  - Iteration
- Today we'll learn
  - Sequence – executing multiple lines of code in order
  - Selection – the "if" statement
  - Iteration – the "for"statement

Keywords

- Keywords are in lower case.
- Keywords are words that you can't use in your application (without some extra work).
- Keywords are defined in the language specification.
- The editor shows keywords in blue. This is configurable.

Variables

- Variables are values that are stored in memory.
  - Variables have names.
  - Variables can be **set**, meaning their values can be changed.
  - Variables can be **read**, meaning their value can be retrieved.
  - Variables have **type** and must be **declared**.
    - Many types are defined in the Framework. We'll work with integers and strings this week.

Expressions

- An expression is constructed from operators and operands.
  - `1 + 2`
  - Total * 0.08
- An expression returns a value.
- The spec has a complete list of operators.

Statements

- "The actions of a program are expressed using statements." (spec)
- Examples of statements include:
  - Declaration
  - Assignment (a variation on an expression statement)
  - Selection
  - Iteration
- The spec has a complete list of statements.

Console Input

- The Console.ReadLine method **returns** all the characters typed by the user, up to the carriage return.
- The **return value** is **assigned** to a **string variable** using an assignment statement.

DEMO: Console Input

- Create a new console application.
- Add this code to echo input back to the user.

```
// Read in and print one line from the user.
Console.WriteLine("Enter text:");
string oneLine = "";
oneLine = Console.ReadLine();
Console.WriteLine(oneLine);
```

- Experiment using Write instead of WriteLine.

The string Type

- Press F1 on "string" to learn more about this class.
- The String class has lots of methods for manipulating text.

DEMO: Replacing Txt

- Add this to your code before the last WriteLine method:

```
string output = "";
output = oneLine.Replace("hello", "goodbye");
```

- Change the last line of code to:

```
Console.WriteLine(output);
```

Fun Things to Do with Strings

- Use F1 and IntelliSense to explore the class members.
- Assign to a string literal.
- Concatenation using the + operator.
- Convert to upper case with String.ToUpper.
- Replace substrings with String.Replace.
- Test for equality with ==.

The int type

- The integer type is used to do arithmetic calculations with integers.
  - Integers are different than real numbers (the double type).
- DEMO: Create a new console application and run this code:

```
int one = 1;
int two = 2;
int sum = one + two;
Console.WriteLine(sum);
```

- Notice that Console.WriteLine can write both strings and integers.

Arithmetic Operators

- C# has several arithmetic operators.
- DEMO:
  - Use the language spec or the Help system to find the arithmetic operators.
  - Modify the application to output:
    - The difference of one and two.
    - The square of two (2 times 2).
  - What happens if you divide one by two?

Converting Strings to Integers

- The return value from Console.ReadLine is a string variable.
- You have to use numeric variables to do arithmetic.
- Conversion methods are part of the .NET Framework.

DEMO: Multiplication

- Run this code to add multiply two integers:

```
string input1 = Console.ReadLine();
string input2 = Console.ReadLine();
int number1 = int.Parse(input1);
int number2 = int.Parse(input2);
int product = number1 * number2;
Console.WriteLine(product);
```

The bool type

- The **bool** type holds only two values – **true** and false.
- **true** and **false** are keywords in C#.

DEMO: If Statement

- Create a new console application and run this code:

```
Console.WriteLine("Enter red or blue:");
string answer = Console.ReadLine();
if (answer == "red")
{
   Console.ForegroundColor = ConsoleColor.Red;
}
else
{
   Console.ForegroundColor = ConsoleColor.Blue;
}
Console.WriteLine(answer);
Console.ResetColor();
```

The "if" Statement

- The if statement lets you test a **condition** and execute code depending on the result of the test.
- You can use the == (equals) operator for testing.
  - The == operator returns a **bool** value of either **true** or **false**.
  - In an if test, there is no explicit bool variable declaration.
- You use curly braces to indicate the code to be executed.

If Statement

- You can put more than one statement in each **block**.
  - The block is defined by the curly braces.
- "if" and "else" are keywords.

DEMO: For Statement

- Create a new console application and run this code:

```
for (int count = 0; count < 10; count++)
{
  Console.WriteLine(count);
}
```

The "for" statement

- The for statement lets you execute a block of code a certain number of times.
- The statement has three parts:
  - A variable declaration used for counting (initializer).
  - A test for when the count is complete (condition).
  - An expression to increment the counting variable (iterator).

For Statement

- The syntax or grammar, from the spec is:
  - for (*for-initializer*$_{opt}$; *for-condition*$_{opt}$; *for-iterator*$_{opt}$) *embedded-statement*
- Initializer – int count = 0
- Condition – count < 10
- Iterator – count++
  - Count++ is an expression statement
- Statements – delimited by {}

Writing Condition Expressions

- You use the relational and equality operators to create the condition express:
  - < > <= >= == !==
- These operators compare two values and return a bool value.

Fun Things to Do with Code

- Nesting – You can put a for statement inside a for statement.
- Chaining.
- Using an expression as a method parameter.
- Calling multiple methods in a console application.
- Assign a new value to an existing variable.

Introduction to Scope

- Scope – You can declare variables outside the for loop or if statement and access them within.

Homework 1

- Using only the techniques in this lecture, my solutions to the homework problems were this many lines or less, including lines with braces:
  - Squares – 15 lines
  - Totals – 10 lines (with no error checking)

Turning In Homework

- Two ways to turn in an assignment
  - Upload your assignment to the course website, or
  - Demonstrate your solution to me in class
- To upload your assignment
  - Create a zip file archive of your assignment directory
  - Upload the zip file archive to the course website
- To demonstrate your solution in class
  - I'll come by and check your code.
  - You don't have to get everything right before you leave, but I encourage you to finish at home.

Reading

- C# Language Specification:
  - 1.1 - 1.5
  - The purpose is not to memorize the specification. The purpose is to be aware of the options available in the language and how the language is structured.
- C# for Programmers
  - Chapters 1, 2, 3

Variable Summary

- String summary

```
string name;                  // declaration
string city = "Seattle";      //declaration and initialization
Console.WriteLine("Boston");// write a string literal
Console.WriteLine(city);      // write a string variable
name = "smith";               // assign a string literal to a string variable
name = city;                  // assign a string variable to a string variable
```

- Integer summary

```
int cats;                     // declaration
int dogs = 4;                 // declaration and initialization
Console.WriteLine(4);         // write an integer literal
Console.WriteLine(dogs);      // write an integer variable
cats = 3;                     // assign a integer literal to a integer variable
cats = dogs;                  // assign a integer variable to a integer variable
int pets = int.Parse("45"); // convert a string literal to an integer
pets = cats + dogs;           // sum two integers and assign to an integer
```

For Loop Summary

```
// Write 1 to 10
for (int count = 0; count < 10; count++)
{
    Console.WriteLine(count);
}

// write 1 to howMany, in this case, 4
int howMany = 4;
for (int index = 0; index < howMany; index++)
{
    // Add statements here.
}
```

If Summary

```
// only do something if answer is "yes"
string answer = "yes";
if (answer == "yes")
{
    // Add code here
}

// do something different if answer is "yes" or not
if (answer == "yes")
{
    // Add code here
}
else
{
    // Add code here
}
```