COIS1020H: Programming for Computing Systems

using
Microsoft Visual C# 2010/12

Chapter 1
A First Program Using C#

Programming

- Computer **program**
  - Set of instructions that tells a computer what to do
  - Also called software
- Software comes in two broad categories
  - **System software**
  - **Application software**
- **Machine language**
  - Expressed as a series of 1s and 0s
    - 1s represent switches that are on, and 0s represent switches that are off
    - Understandable to a computer

Programming (cont'd.)

- **High-level programming languages**
  - Use reasonable terms such as “read,” “write,” or “add”
    - Instead of the sequence of on/off switches that perform these tasks
  - Has its own **syntax** (rules of the language)
- **Compiler/Interpreter**
  - Translates high-level language statements into machine language

3

Programming (cont'd.)

- **Programming logic**
  - Involves executing the various statements and procedures in the correct order
    - To produce the desired results
- **Debugging**
  - Process of removing all syntax and logical errors from the program

4

Procedural and Object-Oriented Programming

- **Procedural program**
  - Create and name computer memory locations that can hold values (**variables**)
  - A series of steps or operations to manipulate those values
- **Procedures/methods**
  - Logical units that group individual operations in a program
  - **Called** or **invoked** by other procedures/methods

5

Procedural and Object-Oriented Programming (cont'd.)

- **Object-oriented programming**
  - An extension of procedural programming
- **Objects**
  - Contain their own variables and methods
  - **Attributes/Fields** of an object represent its characteristics (data)
  - **State of an object** is the collective value of all its attributes at any point in time
  - **Behaviors of an object** are the things it "does"
    - **Methods**

6

Features of Object-Oriented Programming Languages

- **Classes**
  - Template from which objects are created
  - Defines the attributes and methods of every object that is an **instance** of that class
- **Objects**
  - An instance of a class
- **Encapsulation**
  - Technique of packaging an object's attributes and methods into a cohesive unit; undivided entity
    - **black box**

7

The C# Programming Language

- Developed as an object-oriented language
- Part of Microsoft Visual Studio
  - Thus, contains a simple GUI interface
- Modeled after the C++ programming language
  - However, eliminates some of the most difficult features
- Very similar to Java syntactically

8

Writing a C# Program that Produces Output

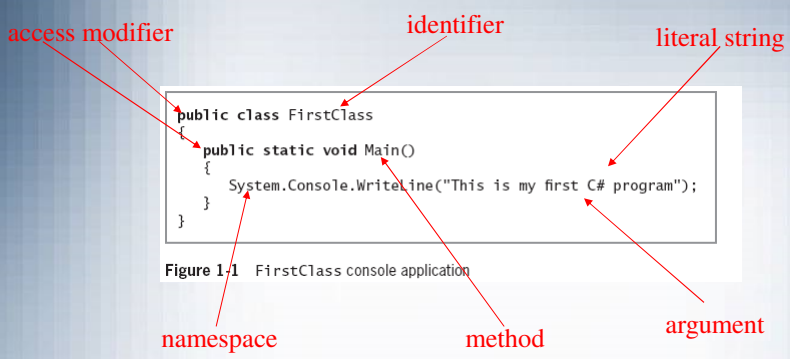

Figure 1-1 FirstClass console application

9

Writing a C# Program that Produces Output (cont'd.)

- **Namespace**
  - Provides a way to group similar classes (more later)
- **C# method parts**
  - **Method header**
    - Includes the method name and information about how information will be shared with the method
  - **Method body**
    - Contained within a pair of curly braces and includes all the instructions executed by the method

10

Writing a C# Program that Produces Output (cont'd.)

- **Whitespace**
  - Any combination of spaces, tabs, and carriage returns (blank lines)
  - Organizes your code and makes it easier to read
- **Access modifier**
  - Defines the circumstances under which the method can be accessed
- **Keywords**
  - Predefined and reserved identifiers that have special meaning to the compiler

11

Writing a C# Program that Produces Output (cont'd.)

- The name of the method is `Main()`
  - Every console application must have a `Main()` method
- The method returns nothing as indicated by the keyword **`void`**

Selecting Identifiers

- **Requirements**
  - Must begin with an underscore, at sign (@), or letter
    - Letters include foreign-alphabet letters
  - Can contain only letters, digits, underscores, and the at sign
    - Not special characters such as #, $, or &
  - Cannot be a C# reserved keyword

13

abstract	float	return
as	for	sbyte
base	foreach	sealed
bool	goto	short
break	if	sizeof
byte	implicit	stackalloc
case	in	static
catch	int	string
char	interface	struct
checked	internal	switch
class	is	this
const	lock	throw
continue	long	true
decimal	namespace	try
default	new	typeof
delegate	null	uint
do	object	ulong
double	operator	unchecked
else	out	unsafe
enum	override	ushort
event	params	using
explicit	private	virtual
extern	protected	void
false	public	volatile
finally	readonly	while
fixed	ref	

Table 1-1 C# reserved keywords

14

Selecting Identifiers (cont'd.)

Class Name	Description
`Employee`	Begins with an uppercase letter
`FirstClass`	Begins with an uppercase letter, contains no spaces, and has an initial uppercase letter that indicates the start of the second word
`PushButtonControl`	Begins with an uppercase letter, contains no spaces, and has an initial uppercase letter that indicates the start of all subsequent words
`Budget2012`	Begins with an uppercase letter and contains no spaces

Table 1-2 Some valid and conventional class names in C#

15

Selecting Identifiers (cont'd.)

Class Name	Description
`employee`	Unconventional as a class name because it begins with a lowercase letter
`First_Class`	Although legal, the underscore is not commonly used to indicate new words in class names
`Pushbuttoncontrol`	No uppercase characters are used to indicate the start of a new word, making the name difficult to read
`BUDGET2013`	Unconventional as a class name because it contains all uppercase letters
`Public`	Although this identifier is legal because it is different from the keyword `public`, which begins with a lowercase "p," the similarity could cause confusion

Table 1-3 Some unconventional (though legal) class names in C#

16

Selecting Identifiers (cont'd.)

Class Name	Description
`an employee`	Space character is illegal
`Push Button Control`	Space characters are illegal
`class`	"class" is a reserved word
`2011Budget`	Class names cannot begin with a digit
`phone#`	The # symbol is not allowed; identifiers consist of letters, digits, underscores, or @

Table 1-4 Some illegal class names in C#

17

Improving Programs by Adding Program Comments

- **Program comments**
  - Nonexecuting statements that document a program
- **Comment out**
  - Turn a statement into a comment
- Types of comments in C#
  - Line comments
  - Block comments

18

Adding Comments to a Program

- Line comment example

```
// Filename Hello.cs
// Written by <your name>
// Written on <today's date>
```

- Block comment example

```
/* This program demonstrates the use of
the WriteLine() method to print the
message Hello, world! */
```

19

Adding Program Comments (cont'd.)

```
/* This program is written to demonstrate
   using comments */
public class ClassWithOneExecutingLine
{
   public static void Main()
   {
      // The next line writes the message
      System.Console.WriteLine("Message");
   }  // End of Main
}  // End of ClassWithOneExecutingLine
```

Figure 1-4 Using comments within a program

20

Using the `System` Namespace

```
public class ThreeLinesOutput
{
   public static void Main()
   {
      System.Console.WriteLine("Line one");
      System.Console.WriteLine("Line two");
      System.Console.WriteLine("Line three");
   }
}
```

Figure 1-5 A program that produces three lines of output

21

Using the `System` Namespace (cont'd.)

```
using System;
public class ThreeLinesOutput
{
   public static void Main()
   {
      Console.WriteLine("Line one");
      Console.WriteLine("Line two");
      Console.WriteLine("Line three");
   }
}
```

Figure 1-7 A program that produces three lines of output with a `using System` clause

22

Writing and Compiling a C# Program

- Steps for viewing a program output
  - Compile **source code** into **intermediate language (IL)**
  - C# **just in time (JIT)** compiler translates the intermediate code into executable statements of machine code
- Can be done using either
  - A **command line environment** like Linux (not for us!)
  - An **Integrated Development Environment (IDE)** like Visual Studio

23

Compiling Code from within the Visual Studio IDE

- Advantages of using the Visual Studio IDE
  - Some of the code you need is already created for you
  - The code is displayed in color
  - You can double-click an error message and the cursor will move to the line of code that contains the error
  - Many other useful debugging tools are available
    - Will be introduced to these in the lectures and labs

24

Example (Part of Lab 1)

- Problem: Create the following C# program (replace xxx with your name) using Microsoft Visual C# 2010

```
using System;
public static class Assign0
{
    public static void Main()
    {
        Console.WriteLine("Hello Rich, my name is xxx");
        Console.ReadLine();
    }
}
```

25

Compiling and Executing a Program Using the Visual Studio IDE (2010)

- Steps
  - Create a new project (Empty Project)
  - Enter the project name
  - Add New Item (Code File)
  - Enter the code file name
  - Write your program using the editor
  - To compile the program, click **Debug** on the menu bar, and then click **Build Solution** (or press **F6)**
  - Click **Debug** on the menu bar and then click **Start With Debugging** (or press **F5**)
    - Output Window Disappears (add **Console.ReadLine();**)
  - Or to **Start Without Debugging** press **CTRL-F5** and Output Window stays until you hit a key to continue

26

Compiling and Executing a Program Using the Visual Studio IDE (cont'd.)

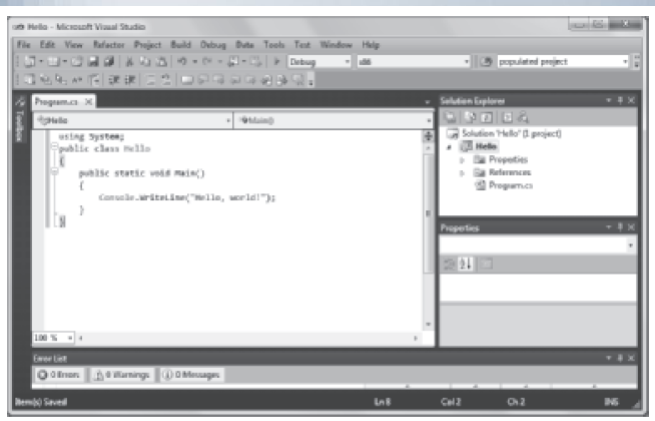

Figure 1-18 The `Hello` application in the IDE

27

Compiling and Executing a Program Using the Visual Studio IDE (cont'd.)

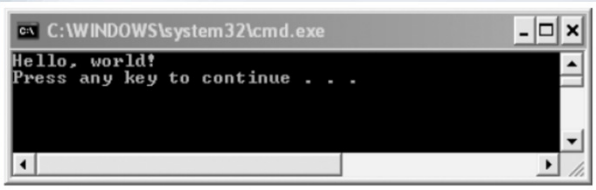

Figure 1-20 Output of the `Hello` application in Visual Studio

28

Summary

- A computer program is a set of instructions that tell a computer what to do
- Understand differences between procedural programming and object-oriented programming
- Objects are instances of classes and are made up of attributes and methods
- The C# programming language is an object-oriented and component-oriented language
- `System.Console.WriteLine()` method
  - Produces a line of console output

29

Summary (cont'd.)

- You can define a C# class or variable by using any name or identifier
- Comments are non-executing statements that you add to document a program
  - Or to disable statements when you test a program
- To create a C# program, use the Microsoft Visual Studio environment

30