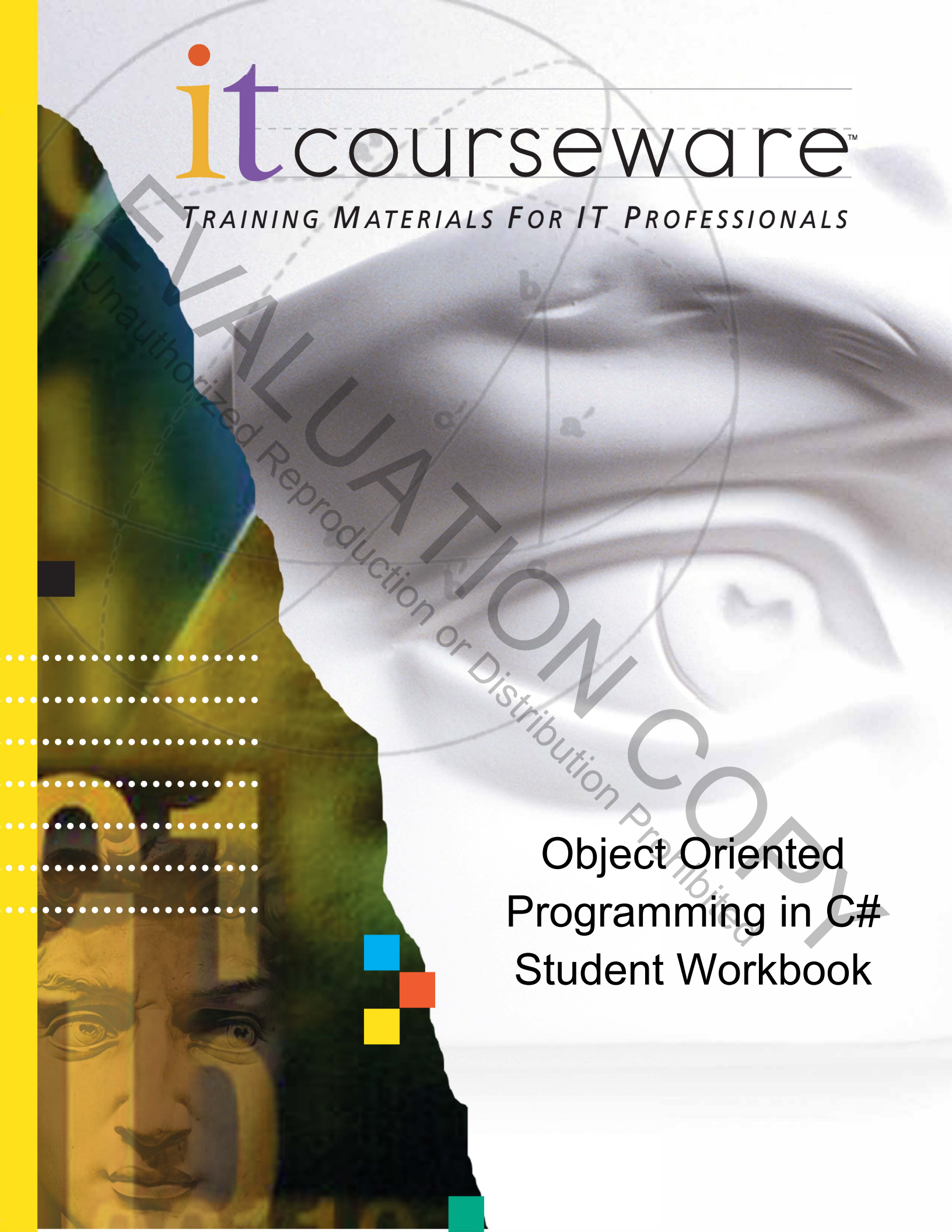


itcourseware™

TRAINING MATERIALS FOR IT PROFESSIONALS

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Object Oriented Programming in C# Student Workbook

Object-Oriented Programming in C#
Rev. 6.1

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Student Guide

Information in this document is subject to change without notice. Companies, names and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Object Innovations.

Product and company names mentioned herein are the trademarks or registered trademarks of their respective owners.

™ is a trademark of Object Innovations.

Authors: Robert J. Oberg, Howard Lee Harkness

Special Thanks: Marianne Oberg, Leslie Koorhan, Robert Canavello, Sharman Staples, Paul Nahay

Copyright ©2022 Object Innovations Enterprises, LLC. All rights reserved.

Object Innovations
877-558-7246
www.objectinnovations.net

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Table of Contents (Overview)

Electronic File Supplements

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright ©2022 Object Innovations Enterprises, LLC
All Rights Reserved

Directory Structure

- **The course software installs to the root directory *C:\OIC\CSharp*.**
  - Example programs for each chapter are in named subdirectories of chapter directories **Chap01**, **Chap02**, and so on.
  - The **Labs** directory contains one subdirectory for each lab, named after the lab number. Starter code is frequently supplied, and answers are provided in the chapter directories.
  - The **CaseStudy** directory contains a case study in multiple steps.
  - The **Demos** directory is provided for performing in-class demonstrations led by the instructor.
  - Supplementary course content is provided in PDF files in the **Supplements** directory. Code examples for the supplements are in directories **Supp1**, **Supp2** and **Supp3**.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright ©2022 Object Innovations Enterprises, LLC
All Rights Reserved

Table of Contents (Detailed)

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright ©2022 Object Innovations Enterprises, LLC
All Rights Reserved

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright ©2022 Object Innovations Enterprises, LLC
All Rights Reserved

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright ©2022 Object Innovations Enterprises, LLC
All Rights Reserved

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright ©2022 Object Innovations Enterprises, LLC
All Rights Reserved

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright ©2022 Object Innovations Enterprises, LLC
All Rights Reserved

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright ©2022 Object Innovations Enterprises, LLC
All Rights Reserved

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright ©2022 Object Innovations Enterprises, LLC
All Rights Reserved

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright ©2022 Object Innovations Enterprises, LLC
All Rights Reserved

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright ©2022 Object Innovations Enterprises, LLC
All Rights Reserved

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright ©2022 Object Innovations Enterprises, LLC
All Rights Reserved

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright ©2022 Object Innovations Enterprises, LLC
All Rights Reserved

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright ©2022 Object Innovations Enterprises, LLC
All Rights Reserved

Chapter 1

Introduction to .NET

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Introduction to .NET

Objectives

After completing this unit you will be able to:

- **Give a high-level overview of .NET, with a focus on .NET 6.0, the basis of .NET implementation going forward.**
- **Outline the architectural components of .NET.**
- **Describe the essentials of creating and running a program in the .NET environment.**
- **Build and run a simple C# program.**
- **Use Visual Studio 2022 as an effective environment for creating C# programs.**
- **Use the .NET documentation.**

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

What is .NET?

- **.NET is Microsoft's software platform for building applications for many environments.**
  - Originally proprietary and restricted to Microsoft Windows, .NET is now open source and cross-platform.
- **.NET applications can be built using multiple languages.**
  - C# and Visual Basic are object-oriented languages.
  - F# is a functional language which also supports object-oriented and imperative programming.
- **.NET Framework is the original implementation of .NET, running on Windows.**
- **.NET Core is a package-based implementation that is cross-platform, running on Mac and Linux besides Windows.**
- **.NET 6 is the successor of .NET Core.**
  - It is the basis of .NET implementation going forward.
- **.NET Framework 4.8 is the latest version of the classical .NET Framework.**
  - It will continue to be distributed with future releases of Windows. As long as it is installed on a supported version of Windows, .NET Framework 4.8 will continue to also be supported.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Libraries and Tools

- **Microsoft and third parties provide many libraries to extend .NET.**
  - Many libraries are furnished through NuGet packages.
  - NuGet is a package manager specifically designed for .NET.
- **Visual Studio is a full-featured IDE running on Windows for building all types of .NET applications.**
  - Visual Studio 2022 is required for .NET 6.
- **Visual Studio for Mac runs on Mac computers.**
  - It can be used for building iOS and macOS.
  - It can also be used for ASP.NET Core apps and services.
- **Visual Studio Code is a lightweight code editor available for Windows, macOS and Linux.**
  - It comes with built in support for JavaScript and extensions for many languages, including C#, C++, Java, Python, PHP, and others.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Application Models

- **Web applications and services**
  - Classical ASP.NET supports web applications on Windows.
  - ASP.NET Core is cross-platform, targeting Windows, macOS and Linux.
- **Mobile applications**
  - Cross-platform targeting iOS, Android and Windows
- **Desktop**
  - Windows desktop applications using Windows Forms or Windows Presentation Foundation (WPF)
  - Mac desktop applications
- **Microservices -- a design pattern in which apps are composed of small, independent modules.**

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Managed Code

- **.NET apps run *managed code*.**
- **Compilers for .NET languages generate *intermediate language* (IL).**
- **An IL program does not run directly on hardware but on a virtual machine, called a *runtime.***
  - The .NET runtime is known as the Common Language Runtime, or CLR.
- **The runtime provides various services such as automatic memory management, type safety, and so on.**

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

.NET Programming in a Nutshell

Write your program in a high-level .NET language, such as C#.

Compile your program into IL.

Run your IL program, which will launch the runtime (CLR) to execute your IL, using its just-in-time compiler to translate your program to native code as it executes.

- **We will look at a simple example of a C# program and run it under .NET.**
  - See **Chap01\SimpleCalc**
  - Here is the code:

```
int width = 20;
int height = 5;
int area;
area = width * height;
Console.WriteLine("width = {0}", width);
Console.WriteLine("height = {0}", height);
Console.WriteLine("area = {0}", area);
```

  - Here is the output when the program is run:

```
width = 20
height = 5
area = 100
```

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Visual Studio 2022

- **While it is possible to write C# programs using any text editor, and compile them with the command-line compiler, it is very tedious to program that way.**
- **An IDE makes the process of writing software much easier.**
  - An IDE provides convenience items, such as a syntax-highlighting editor.
  - An IDE reduces the tedium of keeping track of configurations, environment settings and file organizations.
- **You may use Visual Studio 2022 throughout this course to create and compile your C# programs.**
- **Visual Studio 2022 is discussed in more detail in Supplement 1.**
- **In this course you may use any version of VS 2022, including the free Visual Studio 2022 Community.**

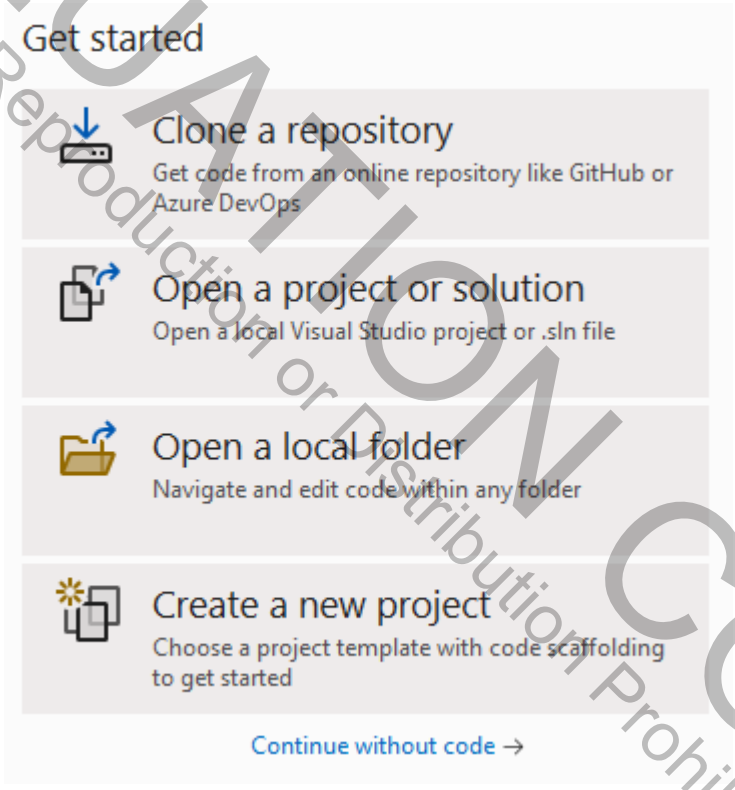
EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Using Visual Studio

- **First, let's see how to open and run an existing program.**

1. From the Visual Studio start page click on "Open a project or solution". This will bring up the New Project dialog. (You can also open a project from the menu File | Open | Project/Solution.)

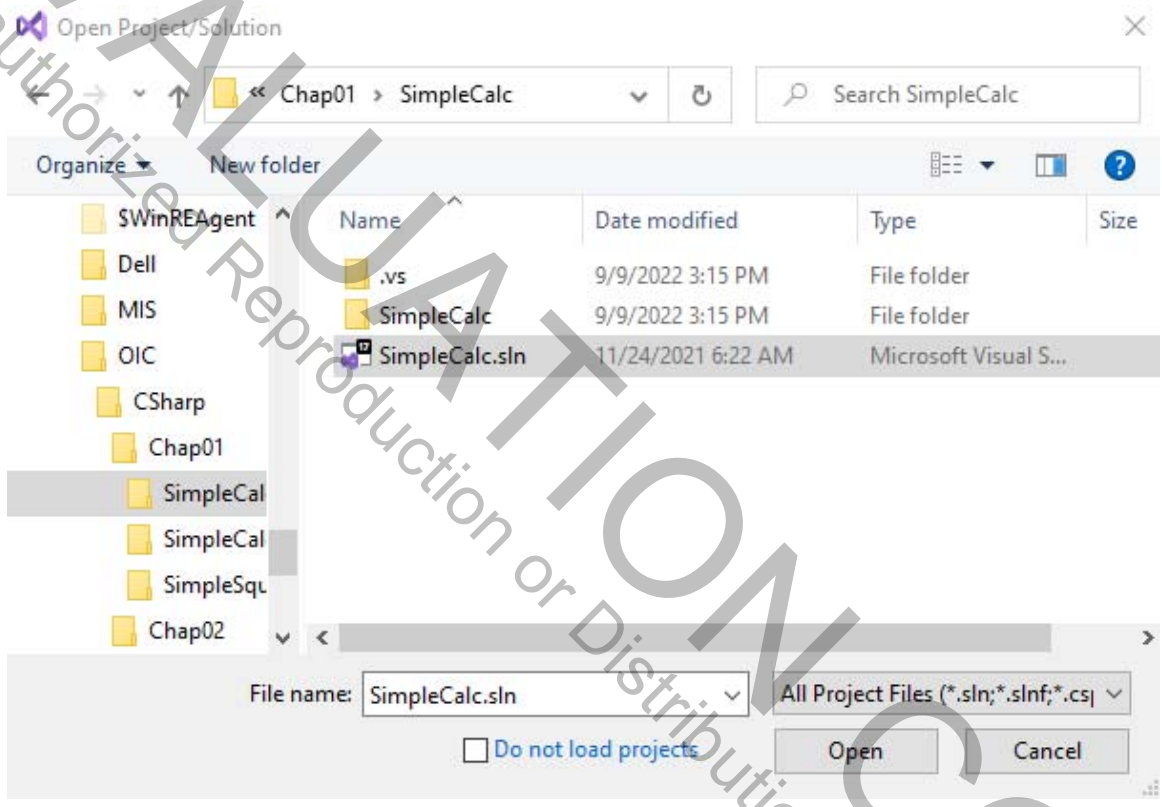


EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Using Visual Studio (Cont'd)

2. This will bring up the Open Project/Solution dialog. Navigate to the **SimpleCalc** folder in **CSharp\Chap01**. Select the solution file **SimpleCalc.sln**.



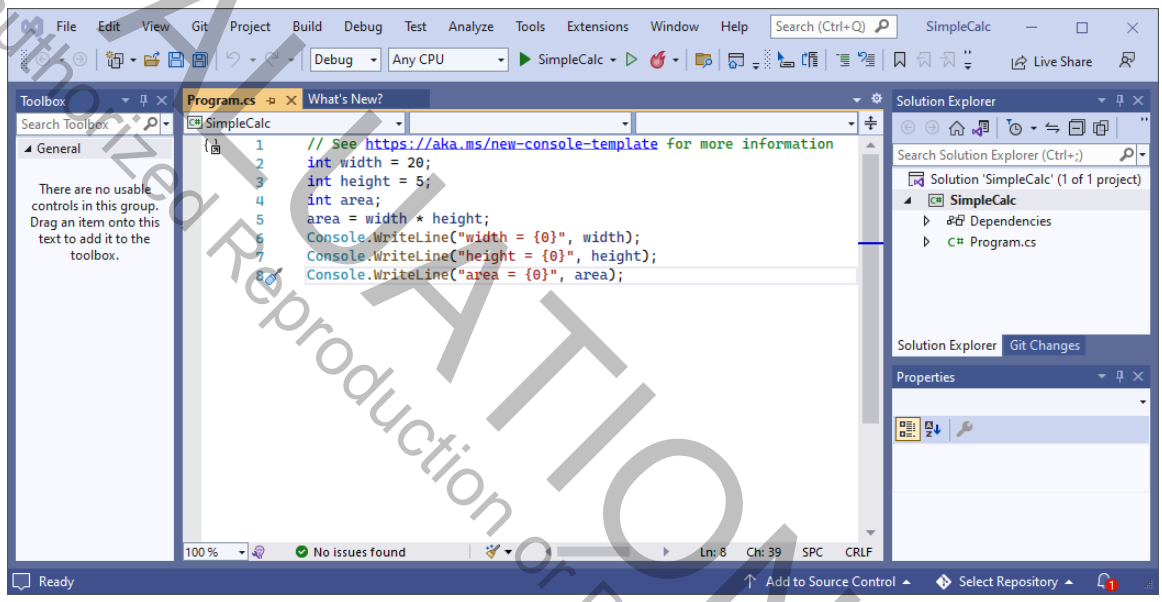
3. Click Open.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

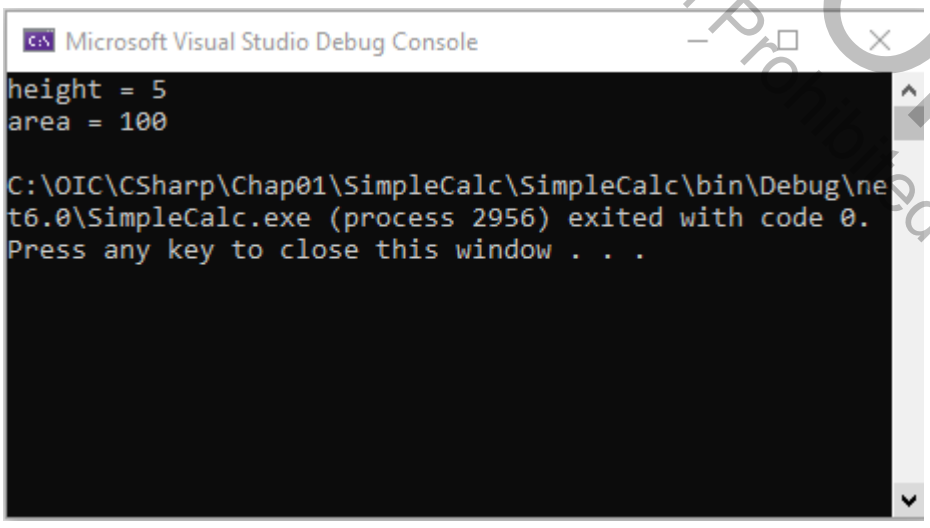
Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Using Visual Studio (Cont'd)

- **In the middle pane you will see the program open in the Visual Studio code editor, and Solution Explorer is on the right.**



- **Now let's run the program, without debugging.**
  - The easiest way to do that is with the open green wedge on the toolbar.



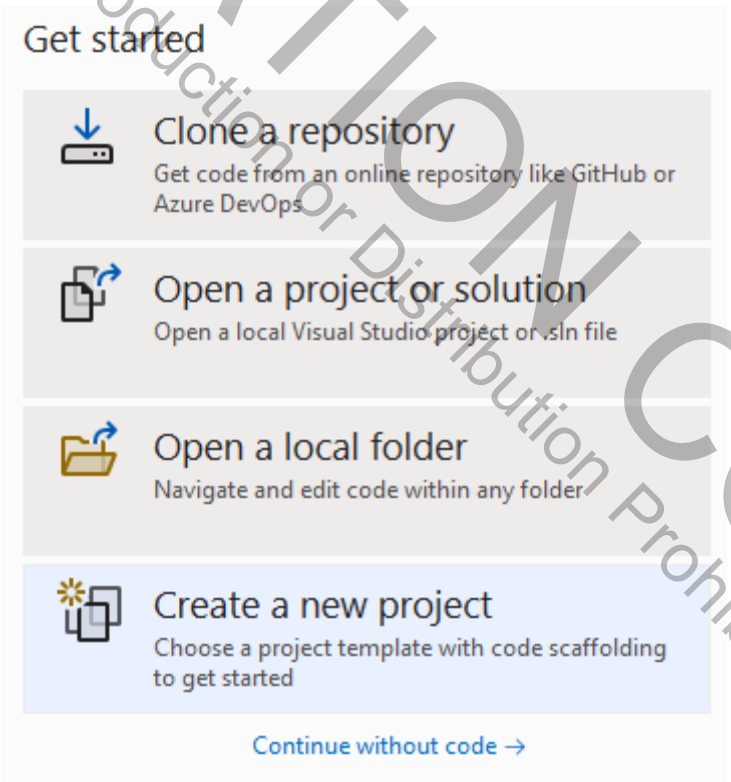
EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Visual Studio Demo

- **We will now create a simple console application using Visual Studio.**
  - Our program will be similar to the example we looked at, only simpler and requiring less typing! It simply calculates the area of a square.

1. From the Visual Studio start page click on "Create a new project". This will bring up the New Project dialog. (You can also use the menu File | New | Project when the main Visual Studio window is open.)

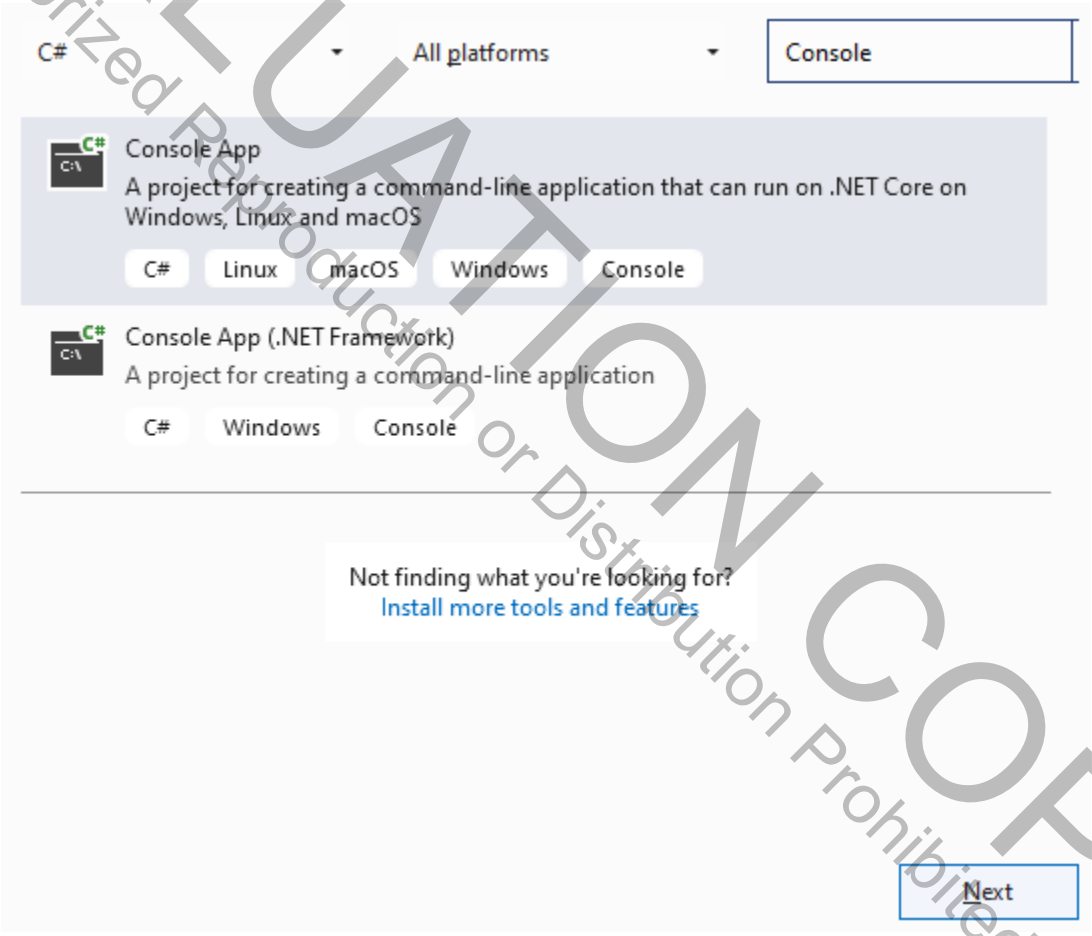


EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Creating a Console App (Cont'd)

2. Choose Console App with language C#. We filtered the language to be C# and the project type to be Console.
   - Note that Console App (.NET Framework) would generate an app using the classical .NET Framework that runs only on Windows.



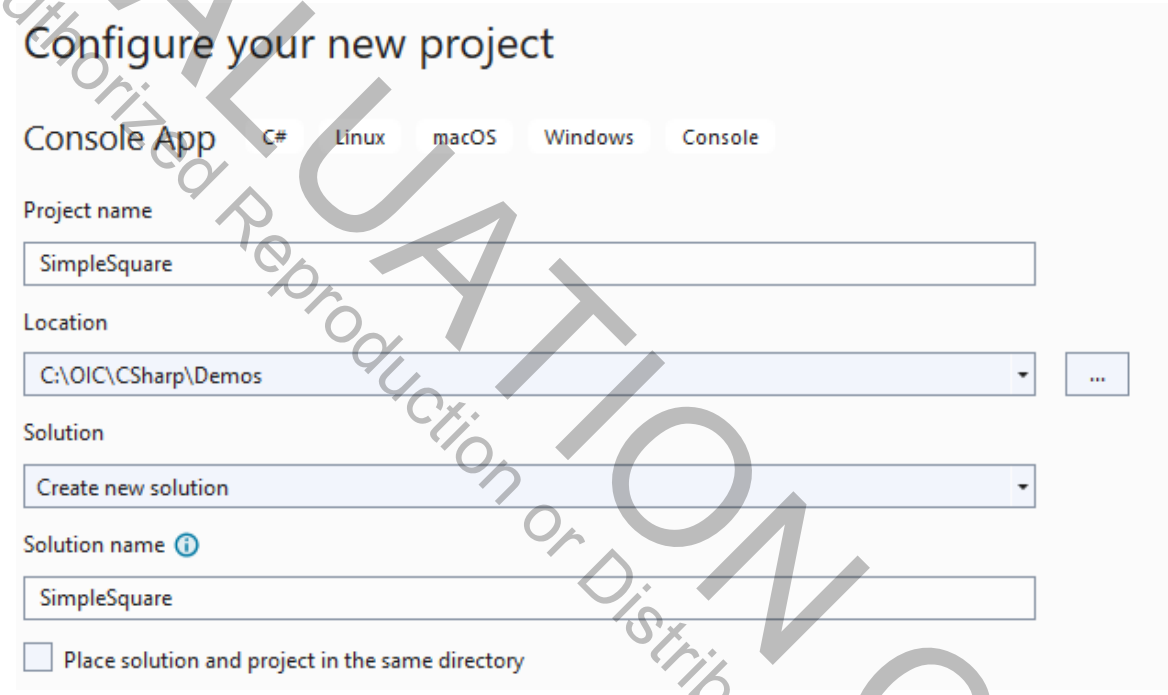
3. Click Next.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

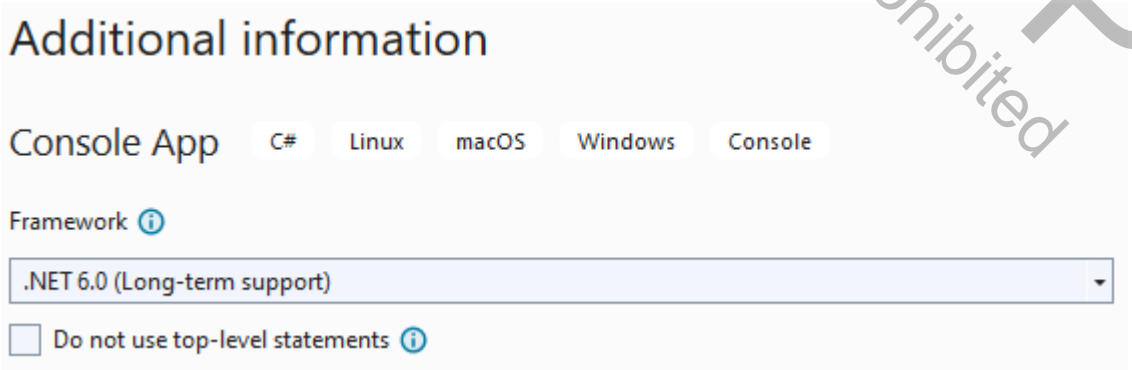
Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Configure Your New Project

4. In the Project name field, type **SimpleSquare** and for Location browse to **C:\OIC\CSharp\Demos**. Leave Solution name as SimpleSquare. Leave unchecked "Place solution and project in the same directory."



5. Click Next. Under “Additional Information” accept the suggested .NET 6.0 (Long-term support) for Framework. Leave unchecked “Do not use top-level statements”.



EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

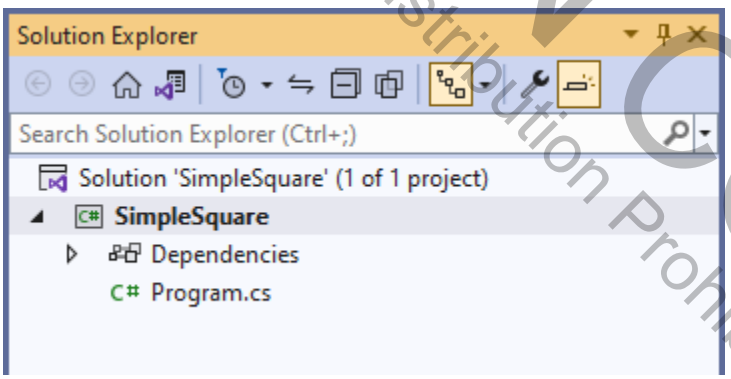
Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Editor and Solution Explorer

- **You will see the file *Program.cs* displayed in an editor window.**
  - If you are familiar with .NET from earlier versions, you are in for a big surprise!

  - There is a new template for console programs. You can follow the link for more information, and we will discuss it in the next chapter.

- **On the right you will see Solution Explorer open.**



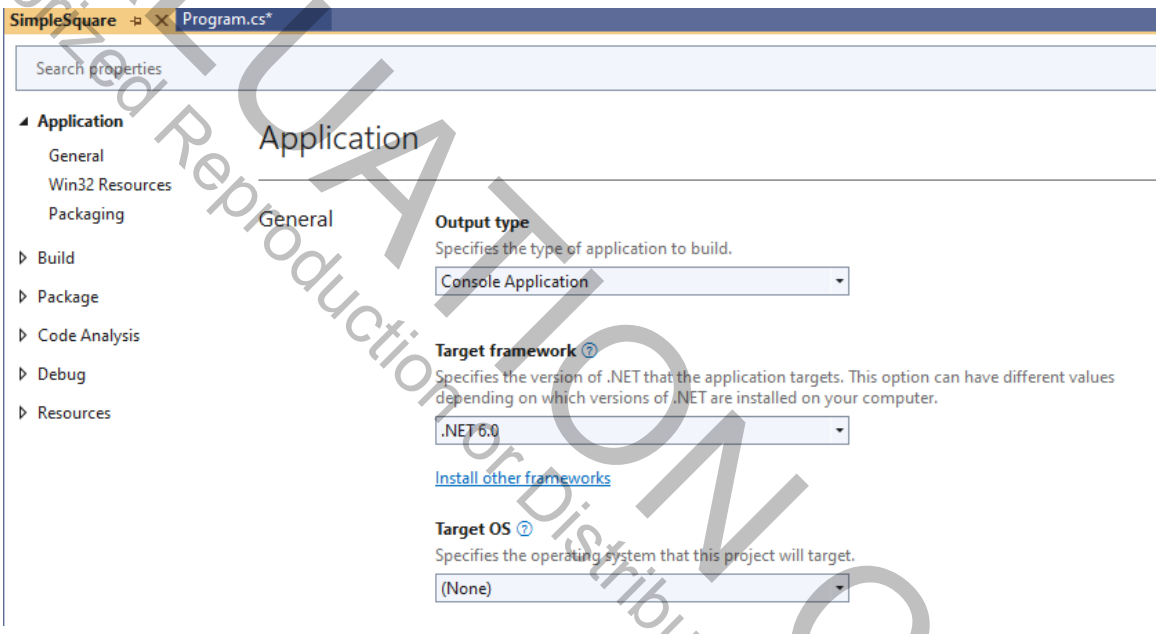
EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Project Properties

– Right-click over the **SimpleSquare** project (shown highlighted in the screen capture) and select Properties at the bottom of the context menu.

– Observe that .NET 6.0 is the Target framework, under the Application | General properties.



– Don't worry about the other project properties for now.

– The Properties window is set up to work with the build system and to have convenient default properties.

– Close the project file and return to **Program.cs**, in the Visual Studio code editor.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

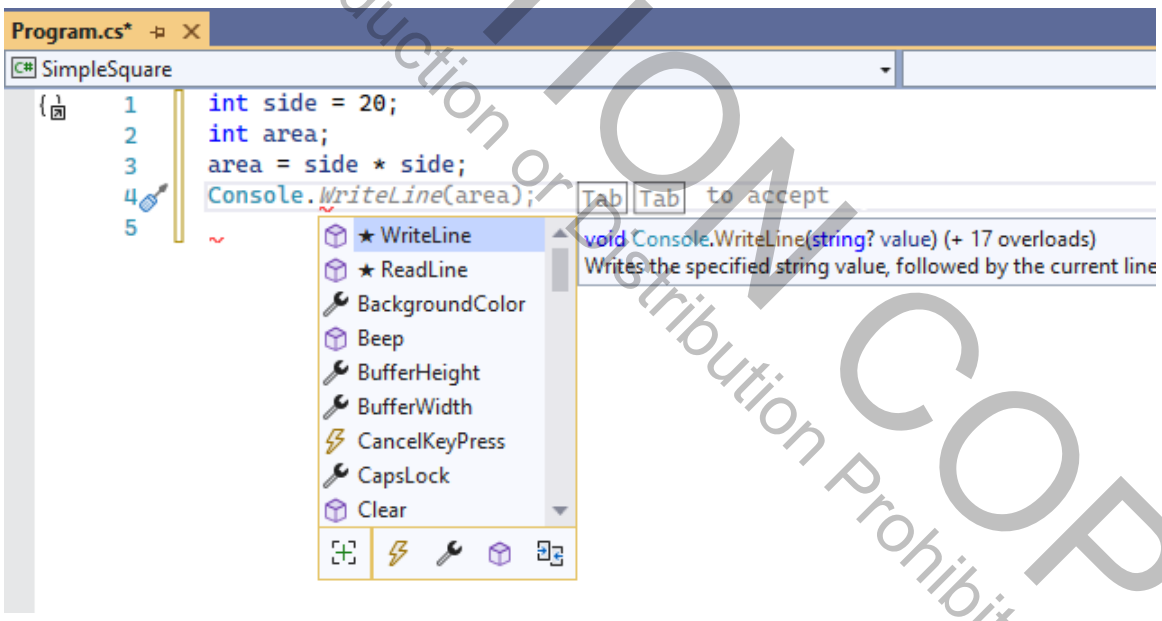
Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Visual Studio Intellisense

- **The file *Program.cs* will be open in the Visual Studio text editor. Replace the starter code by the following, which you can type in.**

```
int side = 20;
int area;
area = side * side;
Console.WriteLine("side = " + side);
Console.WriteLine("area = " + area);
```

- **Notice what happens as type *Console* followed by a period.**



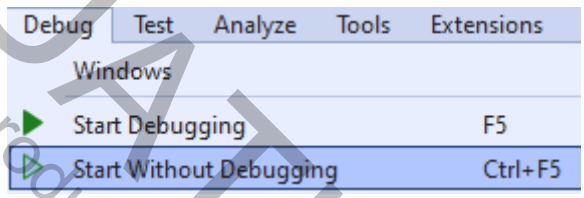
- **A powerful feature of Visual Studio is *IntelliSense*.**
  - IntelliSense will automatically pop up a list box allowing you to easily insert language elements directly into your code.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

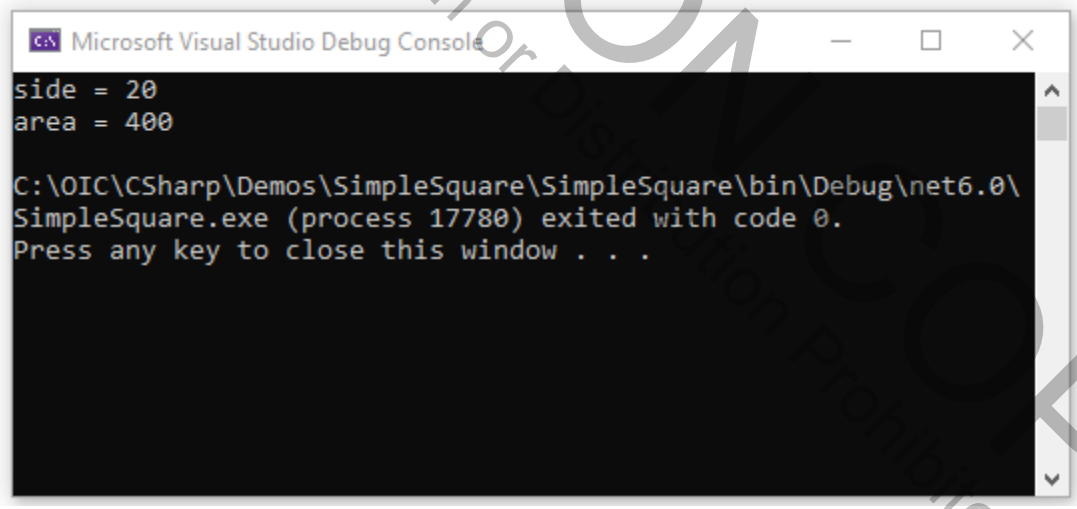
Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Running the Program

- **After you have typed in the complete program, you can run it without debugging in several ways:**
  - Using the open green wedge on the toolbar 
  - Using the keyboard shortcut Ctrl + F5
  - From the menu Debug | Start Without Debugging



- **This is the result:**

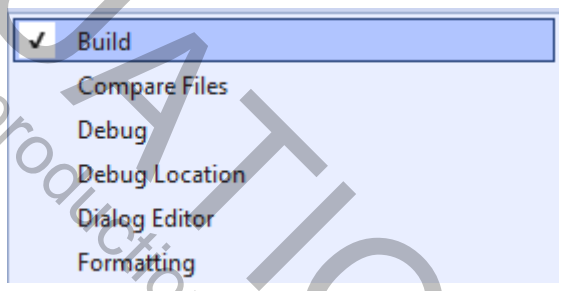


EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Build the Project

- **Building a project means compiling the individual source files and linking them together with any library files to create an IL executable .EXE file.**
- **To make it easier to build, add the Build toolbar (if it is not already present) by a right-click over the toolbar area. Check Build.**



  - This will add the Build toolbar to the Standard and Text Editor toolbars that are shown by default.

- **Then you can build the project by using one of the following:**
  - Menu Build | Build Solution or toolbar button or keyboard shortcut Ctrl+Shift+B. (This builds all the projects in the solution.)
  - Menu Build | Build SimpleSquare or toolbar button . (This just builds the project SimpleSquare)[1].
- **Both forms of Start (see next page) will automatically build the project if not already built.**

[1] The two are the same in this case, because the solution has only one project, but some solutions have multiple projects, and then there is a difference.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Run the Project

- **You can run the program without the debugger by using one of the following:**
  - Menu Debug | Start Without Debugging
  - Toolbar button  (This button is provided by default in Visual Studio 2022.)
  - Keyboard shortcut Ctrl + F5

- **You can run the program in the debugger by using one of the following:**
  - Menu Debug | Start Debugging
  - Toolbar button 

  - Keyboard shortcut F5.
- **We will discuss debugging later.**

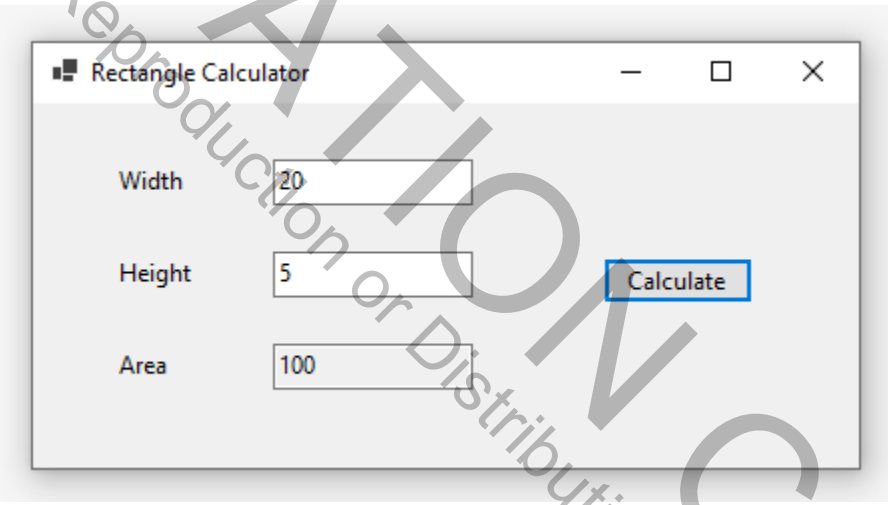
EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Visual C# and GUI Programs

- **Microsoft's implementation of the C# language, Visual C#, works very effectively in a GUI environment.**
  - Using Windows Forms (available in .NET Core with .NET 5.0 and above), it is easy to create Windows GUI programs in C#.

  Example: See **Chap01\SimpleCalcGui**



- **We will discuss GUI programming using C# in Chapter 19.**

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

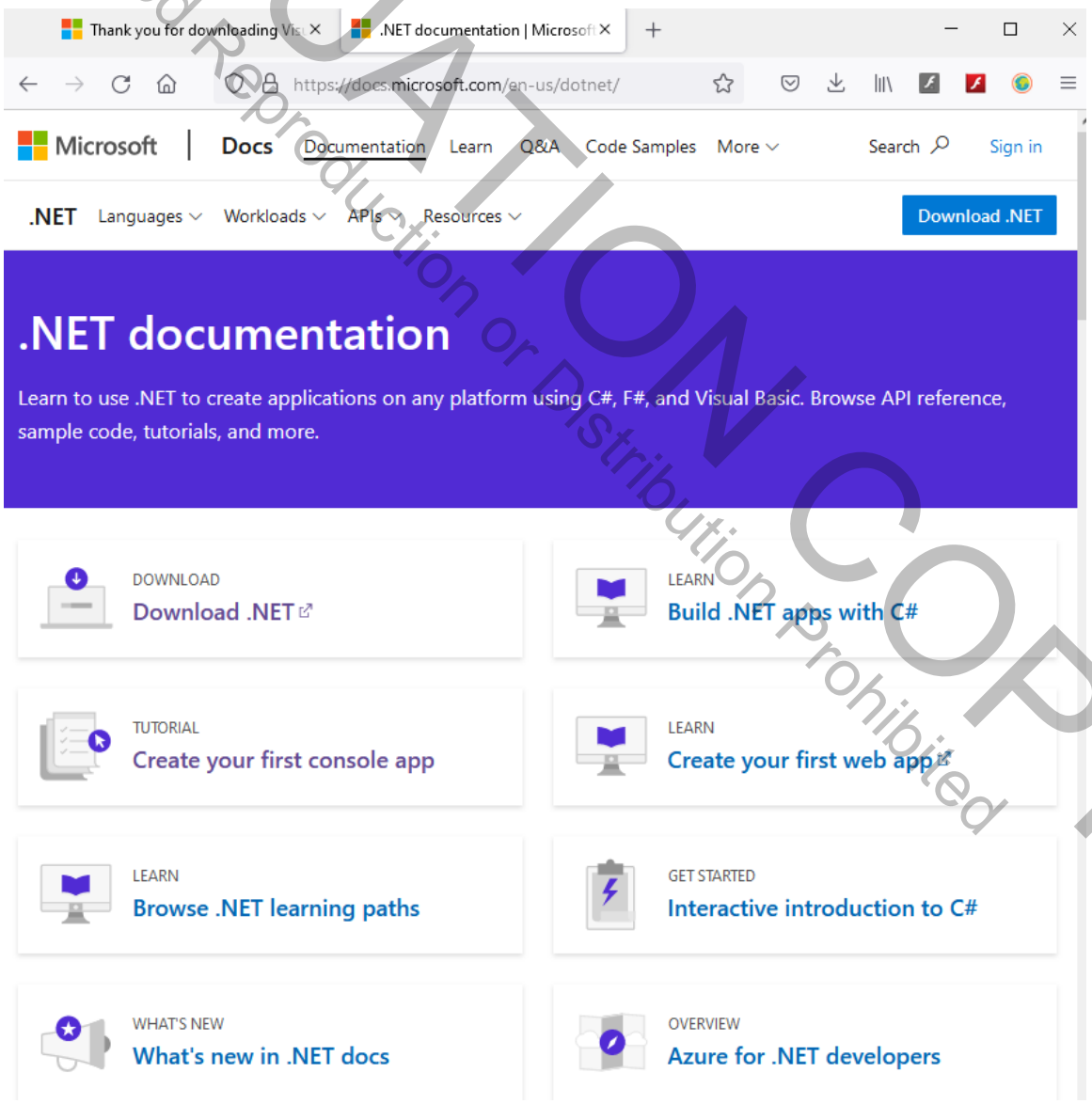
Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

.NET Documentation

- **.NET documentation is available online.**
- **It is now part of comprehensive Microsoft documentation at**

https://docs.microsoft.com

  - Select .NET.



EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Summary

- **.NET is Microsoft's software platform for building applications for many environments.**
- **.NET applications can be built using multiple languages, including C#.**
  - C# is an object-oriented language designed by Microsoft for .NET.
- **The basis of .NET going forward is .NET 6.0, the most recent version of the cross-platform .NET Core.**
- **The classical .NET Framework 4.8 is being maintained.**
- **Visual Studio is a full-featured IDE running on Windows for building all types of .NET applications.**
- **With Visual Studio it is easy to create and run programs using C#.**
- **Windows Forms is supported in .NET 6.0 as an easy means of creating GUI programs running on Windows.**
- **You can access extensive .NET documentation online.**

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Chapter 2

First C# Programs

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

First C# Programs

Objectives

After completing this unit you will be able to:

- **Write a basic "Hello, World" program in C# with and without top-level statements.**
- **Explain the difference between the old and the new console template.**
- **Describe the basic structure of C# programs.**
- **Describe how related C# classes can be grouped into namespaces and the purpose of the *using* statement.**
- **Use variables and simple expressions in C# programs.**
- **Write C# programs that can perform simple calculations.**
- **Perform simple input and output in C#.**
- **Describe objects and classes in C#.**
- **Use an input wrapper class to perform input in C#.**

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Hello, World

- **Whenever learning a new programming language, a good first step is to write and run a simple program that will display a single line of text.**
  - Such a program demonstrates the basic structure of the language, including output.
  - You must learn the pragmatics of compiling and running the program.
- **When you create a new C# Console application Visual Studio will create a Hello World program for you!**
  - We have added our own comment to the starter code.

```
// See https://aka.ms/new-console-template
// for more information
// Hello World demo
Console.WriteLine("Hello World!");
```

  - This uses the new console template, which is the default with Visual Studio 2022. You can click on the link for more information.
  - See **Hello\New** in the **Chap02** directory**.**

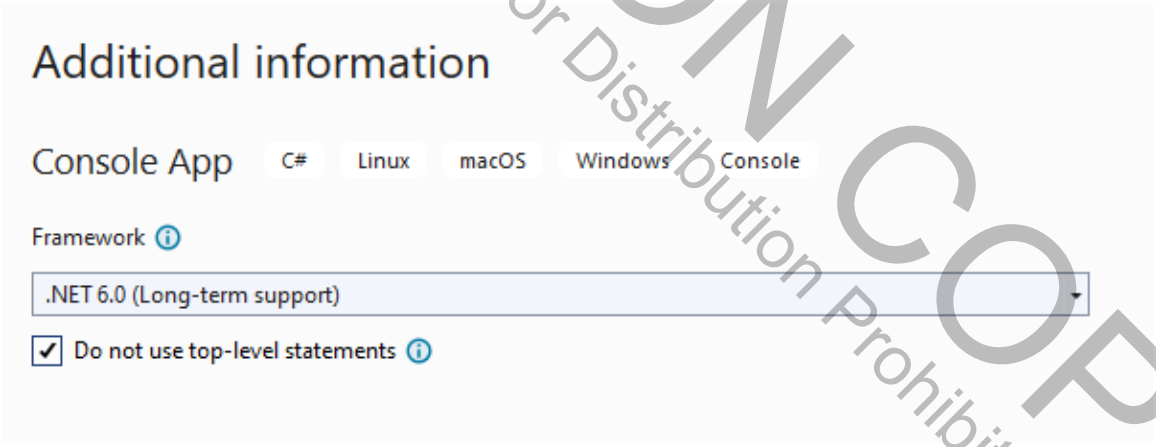
EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Using the Old Console Template

- **Now let's create the same Hello World program without using top-level statements.**
  - We'll use Visual Studio 2022 just like in the previous chapter, but with one difference along the way.

1. From the Visual Studio start page click on "Create a new project". If Visual Studio is already running, use the menu File | New | Project …
2. Choose Console App.
3. Browse to **Demos** for the location and type **Hello** as the project name. Click Next.
4. In the next window check "Do not use top-level statements".



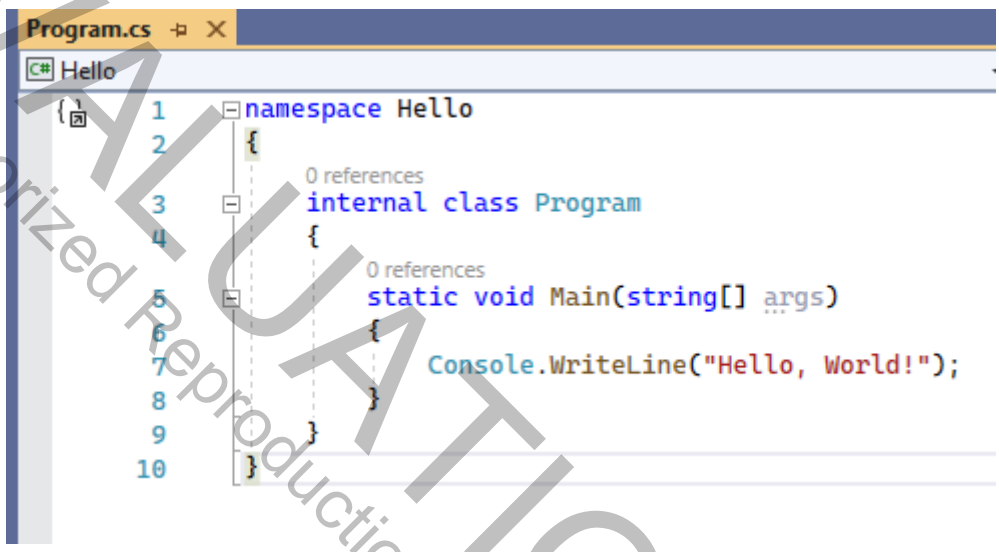
5. Click Create.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Old Console Template (Cont'd)

6. Examine **Program.cs** in the project that Visual Studio created for you



   - This program has several additional program elements, which we will examine in this course. For now, we will focus on the key elements of program structure.
   - Again, we will provide a comment.

```
// Hello World demo
```

   - This version of the Hello World program is saved in **Chap02\Hello\Old**.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Program Structure

```
// Hello World demo
...
    internal class Program
    {
        ...
    }
...
```

- **Every C# program has at least one *class*[1].**
  - A class is the foundation of C#'s support of object-oriented programming.
  - A class encapsulates data (represented by **variables**) and behavior (represented by **methods**).
  - All the code defining the class (its variables and methods) will be contained between the curly braces.
  - We will discuss classes in detail later, including use of the modifier **internal**.
- **Note the *comment* at the beginning of the program.**
  - A line beginning with a double slash is present only for documentation purposes and is ignored by the compiler.
- **C# files have the extension *.cs*.**

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

[1] With the New Console Template in Visual Studio 2022 the compiler will generates a class,

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Program Structure (Cont'd)

```
...
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
...
```

- **Every method in C# has one or more *statements*.**
- **A statement is terminated by a semicolon.**
  - A statement may be spread out over several lines.
- **The *Console* class provides support for standard output and standard input.**
  - The method **WriteLine()** displays a string, followed by a new line.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Namespaces

- **Much standard functionality in C# is provided through many classes in the .NET class library.**
- **Related classes are grouped into *namespaces*.**
- **The old console template explicitly shows the namespace in code.**

```
namespace Hello
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello, World!");
        }
    }
}
```

- **Note: For brevity we will sometimes omit the enclosing namespace in code listings. Thus**

```
internal class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");
    }
}
```

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Echo

- **The Hello program illustrated console output.**
- **The Echo program illustrates both console input and output.**
  - The program prompts the user for a name, and then prints out a personalized greeting.
  - See **Chap02\Echo**.
- **This project has two files.**
  - **InputWrapper.cs** defines a wrapper class that simplifies input.
  - **Program.cs** provides the main code, using the new console template.

```
// Echo
//
// Prompts user to enter name and then
// prints out greeting using name

using OI;

InputWrapper iw = new InputWrapper();
string name = iw.getString("Enter your name: ");
Console.WriteLine("Hello, " + name);
```

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Using InputWrapper

- **The bolded statements illustrate how to use the *InputWrapper* class.**
  - Instantiate an **InputWrapper** object **iw** by using **new**.
  - Prompt to obtain input data by calling the appropriate **getXXX()** method.
- **You may examine the code in the file *InputWrapper.cs* if you wish.**
  - But you don't have to understand the implementation of the **InputWrapper** class to use it!
  - That is the case with the .NET class library—the beauty of encapsulation.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Namespaces and Using

- **The fully qualified name of a class is specified by the namespace, followed by a dot, followed by class name.**

```
System.Console
```

- **A *using* statement allows a class to be referred to by its class name alone.**
  - Examine **Chap02\Hello\Vs2019**, a project created originally using Visual Studio 2019 and .NET 5.0.

```
using System;

namespace Hello
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

  - If you comment out the **using** statement, you will get a compiler error. You could correct it by using the fully qualified name of the **Console** class.

```
System.Console.WriteLine("Hello World!")
```

- **.NET 6.0, used in Visual Studio 2022, has a new feature of *implicit using* statements.**
  - Common namespaces are provided automatically.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Namespaces in Your Own Class

- **When you define a class, it is usually a good idea to place it within a namespace.**
  - We did this for the **InputWrapper** class, putting in the namespace **OI**.

```
namespace OI
{
    public class InputWrapper
    {
        public int getInt(string prompt)
        {
            Console.Write(prompt);
            string buf = Console.ReadLine();
            return Convert.ToInt32(buf);
        }
    ...
```

- **Then in a code file where you make use of the class, provide a *using* statement.**
  - See **Program.cs** in **Chap02\Echo**.

```
using OI;

InputWrapper iw = new InputWrapper();
string name = iw.getString("Enter your name: ");
Console.WriteLine("Hello, " + name);
```

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Variables

- **In C#, you can define *variables* to hold data.**
- **Variables represent storage locations in memory.**
- **In C#, variables are of a specific data *type*.**
  - Some common types are **int** for integers and **double** for floating point numbers.
  - You must declare variables before you can use them.
- **A variable declaration reserves memory space for the variable and may optionally specify an initial value.**

```
int kilo = 1024;      // reserves space and assigns
                      // an intial value
int mega;             //reserves space but does
                      // not initialize
```

  - If a variable is not initialized in its declaration, it should be assigned prior to being used.

```
int kilo;
kilo = 1024;
// Now you may use kilo
```

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Expressions

- **You can combine variables and constants (or "literals") via *operators* to form *expressions*.**
- **Examples of operators include the standard arithmetic operators:**

```
+       addition
-       subtraction
*       multiplication
/       division
```

- **Here are some examples of expressions:**

```
kilo * 1024
(fahrenheit – 32) * 5 / 9
3.1416 * radius * radius
```

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Assignment

- **You can assign a value to a variable by using the = symbol.**
  - On the left-hand side is a variable.
  - On the right-hand side is an expression.
  - The expression is evaluated, and its value is assigned to the variable on the left.
  - Assignment is a statement and must be terminated by a semicolon.

```
mega = kilo * 1024;
celsius = (fahrenheit – 32) * 5 / 9;
area = 3.1416 * radius * radius;
```

- **Note that the same variable can be used on both sides of an assignment statement.**

```
int item = 5;
int total = 30;
total = total + item;
```

  - The expression **total + item** evaluates to 35, using the old value of **total**, and this value is assigned to **total**, creating a new value.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Calculations Using C#

- **You can use C# to perform calculations.**
  - Declare whatever variables you need.
  - Create expressions and assign values to your variables.
  - Print out the answer using **Console.WriteLine()**.
- **You can easily perform labeled output, relying on two features of C#:**
  - The operator + performs concatenation for **string** data.
  - There is an automatic, implicit conversion available that converts numeric data to string data when required.
  - Hence this code ...

```
int total = 35;
System.Console.WriteLine("The total is " + total);
```

  - ... will produce this output:

```
The total is 35
```

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Sample Program

- **This program will convert temperature from Fahrenheit to Celsius.**
  - See **ConvertTemp\Step1**[2].

```
// ConvertTemp - Step 1
//
// Program converts a hardcoded temperature in
// Fahrenheit to Celsius

int fahr = 86;
int celsius = (fahr - 32) * 5 / 9;
Console.WriteLine("fahrenheit = " + fahr);
Console.WriteLine("celsius = " + celsius);
```

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

[2] We will usually omit the chapter number in identifying a sample program when it is in the current chapter. This example program is **Chap02\ConvertTemp\Step1**.

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

More about Output in C#

- **The *Console* class in the *System* namespace supports two simple methods for performing output:**
  - **WriteLine()** writes out a string followed by a new line.
  - **Write()** writes out just the string without the new line.

```
int x = 24;
int y = 5;
int z = x * y;
Console.Write("Product of " + x + " and " + y);
Console.WriteLine(" is " + z);
```

  - The output is all on one line:

```
Product of 24 and 5 is 120
```

- **A more convenient way to build up an output string is to use *placeholders* {0}, {1}, etc.**
  - An equivalent way to do the output shown above is:

```
Console.WriteLine("Product of {0} and {1} is {2}",
   x, y, z);
```

  - The program **OutputDemo** illustrates the output operations just discussed.
  - Later in the course we will see how to control formatting of output, and occasionally in examples we will throw in some simple use of formatting.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Input in C#

- **Our first *ConvertTemp* program is not too useful, because the Fahrenheit temperature is hard-coded.**
  - To convert a different temperature, you would have to edit the source file and recompile.
- **What we really want to do is allow the user of the program to enter a value at runtime for the Fahrenheit temperature.**
- **Although simple console input in C# is fairly easy, we can make it even easier using object-oriented programming.**
  - We can encapsulate or "wrap" the details of input in a class.
  - It will be easy to use the wrapper class.
- **We already have seen an example of an *InputWrapper* class in our *Echo* sample program**

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

More about Classes

- **Although we will discuss classes in more detail later, there is a little more you need to know now.**
- **A class can be thought of as a template for creating objects.**
  - An **object** is an instance of a **class**.
- **A class specifies data and behavior.**
  - The data is different for each object instance.
- **In C#, you instantiate a class by using the *new* keyword.**

```
InputWrapper iw = new InputWrapper();
```

  - This code creates the object instance **iw** of the **InputWrapper** class.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

InputWrapper Class

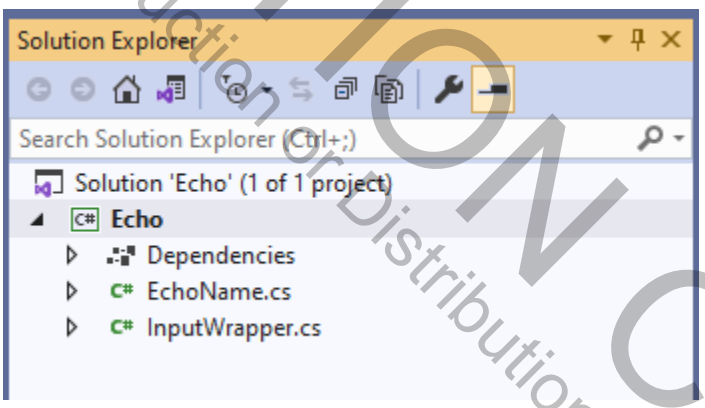
- **The *InputWrapper* class "wraps" interactive input for several basic data types.**
  - The supported data types are **int**, **double**, **decimal** and **string**.
  - Methods **getInt()**, **getDouble()**, **getDecimal()** and **getString()** are provided.
  - A prompt string is passed as an input parameter.
  - See files **InputWrapper.cs** in directory **TestInputWrapper,** which implements the class, and **TestInputWrapper.cs,** which tests the class.
- **Although the code is quite short, it is a little complex, involving a number of different methods from different .NET classes in the .NET Class Library.**
- **However, you do not need to be familiar with the implementation of *InputWrapper* in order to use it.**
  - That is the beauty of "encapsulation"—complex functionality can be hidden by an easy-to-use interface.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Multiple Files in Visual Studio

- **It is very easy to work with Visual Studio projects that have multiple files.**
- **As an example, open the Visual Studio *solution* that is specified by the file *Chap02\Echo\Echo.sln*.**
- **A solution can contain one or more *projects*.**
  - In this case there is a single project file **Echo.csproj**.
- **You can see all the files in a solution through the *Solution Explorer*.**



- **When you build the solution via the toolbar button , you will build all the projects in the solution.**
  - You can also build just the current project via the toolbar button .

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

The .NET Class Library[3]

- **.NET has a very large class library (several thousand classes).**
- **To make all of this functionality more manageable, the classes are partitioned into *namespaces*.**
- **The fundamental namespace is *System*, which contains many useful classes, among them:**
  - **Console** provides access to standard input, output, and error streams for I/O.
  - **Convert** provides conversions among base data types.
  - **Math** provides mathematical constants and functions.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

[3] Traditionally the term ".NET Framework" was used to refer to a collection of class libraries running on Windows. Then Microsoft introduced a new cross-platform implementation built on .NET Core. To avoid confusion, we will use ".NET Framework" to refer to the classical framework running on Windows. We will use the term ".NET Class Library" to refer to Microsoft's class libraries running on .NET Core. This topic is discussed in greater detail in Object Innovations course ".NET Frameworks".

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

The .NET Class Library (Cont'd)

- **Underneath *System*, there are other namespaces, among them:**
  - **System.Data** contains classes constituting the ADO.NET architecture for accessing databases.
  - **System.Xml** provides standards-based support for processing XML.
  - **System.Drawing** contains classes providing GDI+ graphics functionality.
  - **System.Windows.Forms** provides support for creating applications with rich Windows-based interfaces.
  - **System.Web** provides support for browser/server communication.
  - **System.IO** provides support for reading and writing with streams and files. Both synchronous and asynchronous I/O are supported.
  - **System.Net** provides support for several standard network protocols.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Lab 2

C# Programs for Calculation

In this lab you modify or implement several C# programs to perform calculations. You need to perform input (through a wrapper class), perform a calculation, and output the result. Do as many of these exercises as time permits. If you have extra time, do some of the optional experiments suggested in some of the exercises, or make up some experiments on your own.

Detailed instructions are contained in the Lab 2 write-up at the end of the chapter.

Suggested time: 30 minutes

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Summary

- **Every C# application has a class with a method *Main*, which is the entry point into the application.**
- **The *Console* class includes methods for performing output, such as *WriteLine()*.**
- **Expressions in C# are formed from literals, variables, and operators.**
- **With the assignment statement, you can assign a value computed by an expression to a variable.**
- **Input in C# is a little more complicated than output, but you can use a wrapper class that encapsulates the required C# classes and presents a simple programming interface.**
- **.NET has a large class library that is partitioned into namespaces.**

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

Lab 2

C# Programs for Calculation

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Introduction

In this lab, you modify or implement several C# programs to perform calculations. You need to perform input (through a wrapper class), perform a calculation, and output the result. Do as many of these exercises as time permits. If you have extra time, do some of the optional experiments suggested in some of the exercises, or make up some experiments on your own.

Suggested Time: 30 minutes

Root Directory: **OIC\CSharp**

Directories:	**Labs\Lab2\ConvertTemp**	(Exercise 1 work)
	Chap02\ConvertTemp\Step1	(Backup of Exercise 1 starter files)
	Chap02\ConvertTemp\Step2	(Answer to Exercise 1)
	Chap02\TestInputWrapper	(**InputWrapper** class)
	Labs\Lab2	(Exercise 2 work)
	Chap02\Circle	(Exercise 2 answer)

Exercise 1. Fahrenheit to Celsius Conversion

Examine the code of the starter program. Build and run. Notice that the Fahrenheit temperature to be converted is hard-coded. Modify the program to prompt the user for a Fahrenheit temperature, read in the value entered by the user, and print out the result. Make use of the wrapper class **InputWrapper** that was discussed in this chapter.

The starter program uses **int** as the data type for temperatures. An optional[4] experiment is to use **double** as the data type. Could you input the Fahrenheit temperature as an **int** and calculate the Celsius temperature as a **double**?

Exercise 2. Calculate the Area of a Circle

1. Use Visual Studio to create a C# Console App project. Click Next.
2. Assign project name **Circle** and for Location navigate to the **Lab2** folder. Click Next.
3. For Framework choose .NET 6.0 and click Create. This will use the new Console template.
4. Copy the file **InputWrapper.cs** from Exercise 1 into the new project.

[4] In optional exercises we sometimes preview things that are discussed later!

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

5. Delete the starter WriteLine() of “Hello, World!”.

6. Provide C# code to prompt the user for the radius, read in the value entered by the user, calculate the area of the circle, and print out the result. For pi, use the approximation 3.1416. What is an appropriate data type to use for radius and area? (Remember your high school geometry! Area = Pi * radius squared.)

7. As an optional experiment, use the class **Math** (in the namespace **System**) for a more accurate value of pi.

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

Copyright © 2022 Object Innovations Enterprises, LLC
All Rights Reserved

EVALUATION COPY
Unauthorized Reproduction or Distribution Prohibited

7400 E. Orchard Road, Suite 1450 N
Greenwood Village, Colorado 80111
Ph: 303-302-5280
www.ITCourseware.com