Introduction to Computers, the Internet and Visual C#

The chief merit of language is clearness.

—Galen

Our life is frittered away with detail. . . . Simplify, simplify.

—Henry David Thoreau

Man is still the most extraordinary computer of all.

—John F. Kennedy

OBJECTIVES

In this chapter you'll learn:

- Basic hardware, software and data concepts.
- The different types of programming languages.
- The history of the Visual C# programming language and the Windows operating system.
- What cloud computing with Windows Azure is.
- Basics of object technology.
- The history of the Internet and the World Wide Web.
- The parts that Windows 8, .NET 4.5, Visual Studio 2012 and Visual C# 2012 play in the Visual C# ecosystem.
- To test-drive a Visual C# 2012 drawing app.

1.5 Machine Languages, Assembly Languages and High-Level Languages

- Programmers write instructions in various programming languages (such as C#), some directly understandable by computers and others requiring intermediate *translation* steps.

1.5 Machine Languages, Assembly Languages and High-Level Languages

- ***Machine Languages***
  - Any computer can *directly* understand *only* its own machine language, defined by its hardware architecture.
  - Machine languages generally consist of numbers, ultimately reduced to 1s and 0s.
  - The term "code" has become more broadly used and now refers to the program instructions in *all* levels of programming languages.

1.5 Machine Languages, Assembly Languages and High-Level Languages

- ***Assembly Languages and Assemblers***
- Machine language was simply too slow and tedious for programmers to work with.
- Instead, programmers began using English-like *abbreviations* to represent elementary operations.
- These abbreviations formed the basis of assembly languages.
- *Translator programs* called assemblers convert assembly-language programs to machine language quickly.

1.5 Machine Languages, Assembly Languages and High-Level Languages

- ***High-Level Languages, Compilers and Interpreters***
- To speed up the programming process, high-level languages were developed in which single statements could be written to accomplish substantial tasks.
- High-level languages, such as C#, Visual Basic, C++, C, Objective-C and Java, allow you to write instructions that look almost like everyday English and contain commonly used mathematical expressions.
- Translator programs called compilers convert high-level-language code into machine language code.

1.8 C#

- In 2000, Microsoft announced the C# programming language.
- C# has roots in C, C++ and Java.
- C# has similar capabilities to Java and is appropriate for the most demanding app-development tasks, especially for building today's large-scale enterprise apps, and web-based, mobile and "cloud"-based apps.

1.8.1 Object-Oriented Programming

- C# is *object oriented*.
- C# has access to the powerful .NET Framework Class Library—a vast collection of prebuilt classes that enable you to develop apps quickly.

Some key capabilities in the .NET Framework Class Library	
Database	Debugging
Building web apps	Multithreading
Graphics	File processing
Input/output	Security
Computer networking	Web communication
Permissions	Graphical user interface
Mobile	Data structures
String processing	

Fig. 1.3 | Some key capabilities in the .NET Framework Class Library.

1.8.2 Event-Driven Programming

- C# is event driven.
- You'll write programs that respond to user-initiated events such as mouse clicks, keystrokes, timer expirations and—new in Visual C# 2012—touches and finger swipes—gestures that are widely used on smartphones and tablets.

1.8.3 Visual Programming

- Microsoft's Visual C# is a *visual programming language*—in addition to writing program statements to build portions of your apps, you'll also use Visual Studio's graphical user interface to conveniently drag and drop predefined objects like *buttons* and *textboxes* into place on your screen, and label and resize them.
- Visual Studio will write much of the GUI code for you.

1.8.4 An International Standard; Other C# Implementations

- C# has been standardized internationally.
- This enables other implementations of the language besides Microsoft's Visual C#, such as Mono (www.mono-project.com) that runs on Linux systems, iOS (for Apple's iPhone, iPad and iPod touch), Google's Android and Windows.
- You can find the C# standard document at:

  www.ecma-international.org/publications/standards/Ecma-334.htm

1.8.5 Internet and Web Programming

- Today's apps can be written with the aim of communicating among the world's computers.
- As you'll see, this is the focus of Microsoft's .NET strategy.
- In Chapters 23, 29 and 30, you'll build web-based apps with C# and Microsoft's ASP.NET technology.

1.8.6 Introducing async/await

- In most programming today, each task in a program must finish executing before the next task can begin.
- This is called *synchronous programming* and is the style we use for most of this book.
- C# also allows *asynchronous programming* in which multiple tasks can be performed at the *same* time.
- Asynchronous programming can help you make your apps more responsive to user interactions, such as mouse clicks and keystrokes, among many other uses.
- Visual C# 2012's new async and await capabilities simplify asynchronous programming, because the compiler hides much of the associated complexity from the developer.

1.8.7 Other Key Contemporary Programming Languages

- Figure 1.4 summarizes some popular programming languages with features comparable to those of C#.

Programming language	Description
C	C was implemented in 1972 by Dennis Ritchie at Bell Laboratories. It initially became widely known as the UNIX operating system's development language. Today, most of the code for general-purpose operating systems is written in C or C++.
C++	C++, an extension of C, was developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories. C++ provides several features that "spruce up" the C language, but more important, it provides capabilities for *object-oriented programming*. It's often used in apps with stringent performance requirements such as operating systems, real-time systems, embedded systems and communications systems. Visual C++ is Microsoft's version of the language.

Fig. 1.4 | Other programming languages. (Part 1 of 3.)

Programming language	Description
Java	In the 1990s, Sun Microsystems (now part of Oracle) developed the C++-based object-oriented programming language called Java. A key goal of Java is to be able to write programs that will run on a great variety of computer systems and computer-control devices—this is sometimes called *write once, run anywhere*. Java is used to develop large-scale enterprise apps, to enhance the functionality of web servers (the computers that provide the content we see in our web browsers), to provide apps for consumer devices (e.g., smartphones, tablets, television set-top boxes, appliances, automobiles and more) and for many other purposes. Microsoft developed C# as a competitive language to Java.

Fig. 1.4 | Other programming languages. (Part 2 of 3.)

Programming language	Description
Visual Basic	Visual Basic evolved from BASIC, developed in the 1960s at Dartmouth College for introducing novices to fundamental programming techniques. When Bill Gates founded Microsoft in the 1970s, he implemented BASIC on several early personal computers. In the late 1980s and the early 1990s, Microsoft developed the Microsoft Windows **graphical user interface (GUI)**—the *visual* part of the operating system with which users interact. With the creation of the Windows GUI, the natural evolution of BASIC was to **Visual Basic**, introduced by Microsoft in 1991 to make programming Windows apps easier. The latest versions of Visual Basic have capabilities comparable to those of C#.
Objective-C	Objective-C is another object-oriented language based on C. It was developed at Stepstone in the early 1980s and later acquired by NeXT, which in turn was acquired by Apple. It has become the key programming language for the Mac OS X desktop operating system and all iOS-based devices, such as iPods, iPhones and iPads.

Fig. 1.4 | Other programming languages. (Part 3 of 3.)

1.9 Microsoft's .NET

- In 2000, Microsoft announced its .NET initiative, a broad vision for using the Internet and the web in the development, engineering, distribution and use of software.
- .NET permits developers to create apps in *any* .NET-compatible language (such as C#, Visual Basic and many others).
- Part of the initiative includes Microsoft's ASP.NET technology, which is used to create web apps that users interact with via their web browsers.

1.9.1 .NET Framework

- The .NET Framework contains the .NET Framework Class Library, which provides many capabilities that you'll use to build substantial C# apps quickly and easily.
- The .NET Framework Class Library has *thousands* of valuable *prebuilt* classes that have been tested and tuned to maximize performance.

1.9.1 .NET Framework

- You'll learn how to create your own classes, but you should *re-use* the .NET Framework classes when possible to speed up the software development process, while enhancing the quality and performance of the software you develop. (below is an example of "Dilbert reuse"…)

1.9.2 Common Language Runtime

- The Common Language Runtime (CLR) executes .NET programs and provides functionality to make them easier to develop and debug.
- The CLR is a virtual machine (VM)—software that manages the execution of programs and hides from them the underlying operating system and hardware.
- The source code for programs that are executed and managed by the CLR is called *managed code*.

1.9.2 Common Language Runtime (Cont.)

- The CLR provides various services to managed code, such as integrating software components written in different .NET languages, error handling between such components, enhanced security, automatic memory management and more.
- Unmanaged-code programs do not have access to the CLR's services, which makes unmanaged code more difficult to write. (msdn.microsoft.com/en-us/library/8bs2ecf4.aspx)

1.9.2 Common Language Runtime (Cont.)

- Managed code is compiled into machine-specific instructions in the following steps:
  - First, the code is compiled into Microsoft Intermediate Language (MSIL). Code converted into MSIL from other languages and sources can be woven together by the CLR—this allows programmers to work in their preferred .NET programming language. The MSIL for an app's components is placed into the app's executable file—the file that causes the computer to perform the app's tasks.

1.9.2 Common Language Runtime (Cont.)

- When the app executes, another compiler (known as the just-in-time compiler or JIT compiler) in the CLR translates the MSIL in the executable file into machine-language code (for a particular platform).
- The machine-language code executes on that platform.

1.9.3 Platform Independence

- If the .NET Framework exists and is installed for a platform, that platform can run *any* .NET program.
- The ability of a program to run without modification across multiple platforms is known as platform independence.
- Code written once can be used on another type of computer without modification, saving time and money.
- Previously, companies had to decide whether converting their programs to different platforms—a process called porting—was worth the cost.
- With .NET, porting programs is no longer an issue, at least once .NET itself has been made available on the platforms.

1.9.4 Language Interoperability

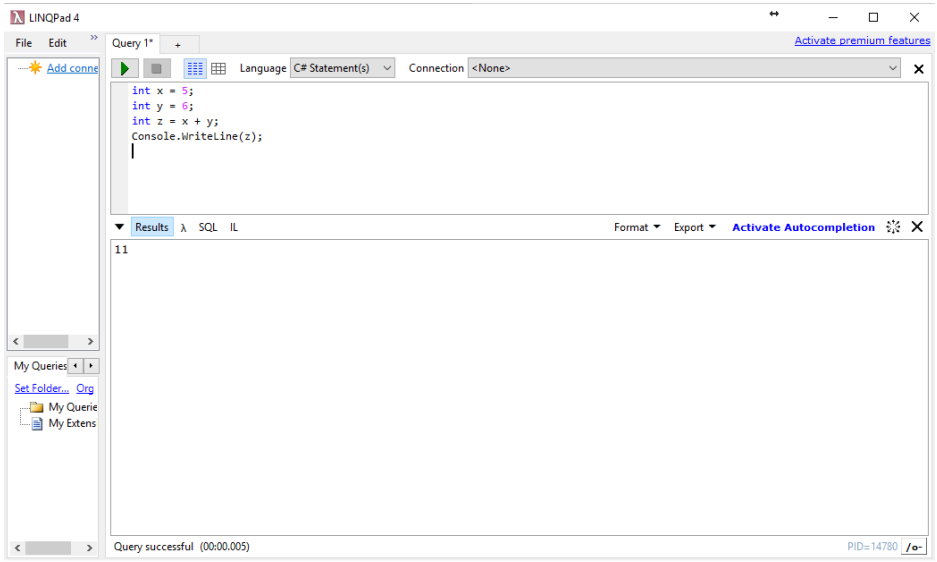
- The .NET Framework provides a high level of language interoperability.
- Because software components written in different .NET languages (such as C# and Visual Basic) are all compiled into MSIL, the components can be combined to create a single unified program.
- Thus, MSIL allows the .NET Framework to be language independent.
- .NET 4.5 features .NET for Windows Store Apps—a subset of .NET that's used to create Windows 8 UI (user interface) style apps.

1.13 Visual Studio Integrated Development Environment

- C# programs are created using Microsoft's Visual Studio—a collection of software tools called an Integrated Development Environment (IDE).
- The Visual Studio 2012 IDE enables you to *write*, *run*, *test* and *debug* C# programs quickly and conveniently.

LinqPad

- In addition to Visual Studio, you should be aware of LinqPad
- LinqPad is a lightweight program that allows you to quickly run some C# code without all of the overhead of a full IDE like Visual Studio



Repl.It

- You should also be aware of an online environment for quick-and-dirty C# (and other languages) coding:
- https://repl.it/

ReSharper

- Another tool you should be aware of is ReSharper from JetBrains
- ReSharper is a plugin for Visual Studio that adds a number of features related to refactoring, searching, intellisence, etc.
- JetBrains has a number of useful tools for developers, and they give free student licenses

Visual Studio Concepts for This Class

- In this class we are going to use **.NET Framework**, NOT .NET Core.
- Visual Studio creates ‘solutions’, a solution is a collection of projects. It is roughly the equivalent of an Eclipse workspace.
- A project is similar to an Eclipse project, a collection of classes that make up your program.
- In this class, we will almost always require only one project per solution, you will create a new solution for EVERY project.
- You must submit the entire SOLUTION, which is the entire directly with the .sln file *in it*. DO NOT submit only the project directory.