# Chapter 1

## Introduction to Computing and Programming

|  |
| --- |
| At a Glance |

#### Instructor’s Manual Table of Contents

* Chapter Overview
* Chapter Objectives
* Instructor Notes
* Quick Quizzes
* Coding Standards
* Resources
* Discussion Questions
* Projects to Assign
* Key Terms

|  |
| --- |
| **Lecture Notes** |

##### Chapter Overview

In this chapter, students review the history of computers and programming languages, including the evolution of C# and .NET. This chapter introduces data and describes how it is represented. Primary types of hardware components are described; differing types of software are discussed. This chapter explains the difference between structured and object-oriented programming and includes the software development methodology used throughout the remainder of the book.

##### Chapter Objectives

In this chapter, students will:

* Learn about the history of computers
* Learn to differentiate between system and application software
* Learn the steps of software development
* Explore different programming methodologies
* Learn why C# is being used today for software development
* Distinguish between the different types of applications that can be created with C#
* Explore a program written in C#
* Examine the basic elements of a C# program
* Compile, run, build, and debug an application
* Create an application that displays output
* Work through a programming example that illustrates the chapter’s concepts

## Instructor Notes

**History of Computers**

Computers have become such an integral part of our lives that many of their functions are taken for granted. Advances in computing are occurring every day, and the programs that distinguish computers have become very complex. To reach this level of complexity, software development has gone through a number of eras.

Computing dates back some 5,000 years. Many consider the abacus to be the first computing device. The Analytical Engine, designed by Charles Babbage and his assistant, Lady Augusta Ada Bryon, Countess of Lovelace, included input devices, memory storage, a control unit that allowed processing instructions in any sequence, and output devices. This was the prototype for what is known today as a general-purpose computer.

Each computing era is characterized by an important advancement. In the mid-1940s, the Second World War, with its need for strategic types of calculations, spurred on the first generation of general-purpose machines. The first era is distinguished by the use of vacuum tubes.

The second generation was characterized by the invention of the transistor in 1956. The software industry (FORTRAN and COBOL) was born.

The third generation, 1964–1971, saw computers become smaller, as transistors were squeezed onto small silicon discs (single chips) that were called semiconductors. Operating systems were first seen in third-generation systems. Many computer historians believe the present day to be either the fourth or fifth generation of modern computing.

During the fourth generation, IBM introduced its personal computer (PC), and in the 1980s, affordable clones of the IBM PC emerged. Defining a fifth generation of systems is somewhat difficult because the generation is still young. Major advances in software are anticipated as Integrated Development Environments (IDEs) such as Visual Studio make it easier to develop applications for the Internet rapidly. Because of the programmability of the computer, the imagination of software developers is free to conjure the computing functions of the future.

**Quick Quiz**

1. True or False: The first generation of computing is characterized by the use of vacuum tubes.  
   Answer: True
2. True or False: The most recognized first computing device is the Pascaline.

Answer: False

1. List characteristics associated with the fifth generation of computing devices.   
   Answer: Computers accept spoken word instructions and imitate human reasoning through artificial intelligence. Computers communicate with devices instantaneously around the globe by transmitting digital media. Social media technologies, cloud computing and the use of mobile applications on smart devices is growing rapidly.
2. The introduction of the IBM PC clones occurred during the \_\_\_\_\_\_\_\_\_\_\_\_\_ generation of computing.

Answer: fourth

**System and Application Software**

Software is the sets of instructions telling the computer exactly what to do. The instructions might tell the computer to add up a set of numbers, compare two names, or make a decision based on the result of a calculation. The two major categories of software are system and application software.

**System Software**

System software is loaded when you power on the computer. Most people associate operating systems with system software. Example operating systems are Windows 8, Android, iOS, Windows 7, and Linux. These types of programs oversee and coordinate the resources on the machine. They include file system utilities, which are small programs that take care of locating files and keeping up with the details of a file’s name, size, and date of creation.

System software programs perform a variety of other functions: setting up directories; moving, copying, and deleting files; transferring data from secondary storage to primary memory; formatting media; and displaying data on screens. Another type of system software includes compilers, interpreters, and assemblers.

**Application Software**

Application software consists of programs developed to perform a specific task. Word processors, such as Microsoft Word, are examples of application software. The programs that you write for this book will be application software.

**Quick Quiz**

1. What are the two types of software?  
   Answer: System and application
2. True or False: Excel is an example of system software. .  
   Answer: False
3. True or False: A compiler is a type of system software.  
   Answer: True
4. Identify two types of functions that are performed by operating system software.  
   Answer: Copy and delete files

**Software Development Process**

A number of different approaches, or methodologies, are used to solve computer-related problems. Successful problem solvers follow a methodical approach with each programming project.

**Steps in the Program Development Process**

1. Analyze the problem. The first step should be directed toward grasping the problem thoroughly. Analyze precisely what the software is supposed to accomplish. During this phase, review the problem specifications, investigate the input and determine what the software should accomplish.
2. Design a solution. Programmers use several approaches, called methods, during design. Procedural and object-oriented methodologies are the two most commonly used design methods.
3. Code the solution. After you have completed the design and verified that the algorithm is correct, you translate the design into source code.
4. Implement the code. During this phase, the typed program statements (source code) are compiled to check for rule violations.
5. Test and debug. Good programmers often build test plans at the same time they are analyzing and designing their solutions. This test plan should include testing extreme values, identifying possible problem cases, and ensuring that these cases are tested.

**Quick Quiz**

1. True or False: A testing plan should be developed early on in the program development process.

Answer: True

1. True or False: The first step in the program development process should be to design a solution.

Answer: False

1. True or False: All programming methodologies should include exactly five steps.

Answer: False

1. Program statements are called \_\_\_\_\_\_\_\_\_\_\_\_\_ code.

Answer: source

**Programming Methodologies**

A methodology is a strategy, a set of steps, or a set of directions to solve a problem. The two most popular programming paradigms used by programmers are structured procedural programming and object-oriented programming.

**Structured Procedural Programming**

Procedural programming is process oriented, meaning it focuses on the processes that data undergoes from input until meaningful output is produced. This approach is very effective for small stand-alone applications. To think algorithmically, programmers use a number of tools. One such tool used is a flowchart. Another tool used to develop an algorithm during design is pseudocode. As the name implies, with pseudocode, steps are written in “pseudo” or approximate code format, which looks like English statements. Structured programming is associated with a technique called top-down design or stepwise refinement. The underlying theme or concept is that given a problem definition, you can refine the logic by dividing and conquering. One of the drawbacks of the procedural approach involves software maintenance. When an application is upgraded or changed, programs written using the procedural approach are more difficult to maintain.

**Object-Oriented Programming**

Viewed as a newer approach to software development, the concept behind object-oriented programming (OOP) is that the focus is on determining the objects you want to manipulate rather than the processes or logic required to manipulate the data. An entity is often defined as a person, place, or thing. It is normally a noun. By abstracting out the attributes or characteristics (data) and the behaviors or actions (processes on the data), you can divide complex phenomena into understandable entities.

A class is like a template; an object is an instance of the class. When these data members are associated with the class, an object is created or constructed. Through inheritance, it is possible to define subclasses of data objects that share some or all of the parent’s class characteristics. This is what enables reuse of code. A class diagram is one of the primary modeling tools used by object-oriented programmers.

Whether you are using a procedural or object-oriented approach, you should follow the five steps to program development. As with the procedural approach, the object-oriented development process is iterative.

**Quick Quiz**

1. Structured approach is \_\_\_\_\_\_\_\_\_\_\_\_\_ oriented.

Answer: process

1. True or False: Characteristics of an entity are also called attributes.

Answer: True

1. An object is a(n) \_\_\_\_\_\_\_\_\_\_\_\_\_ of a class.

Answer: instance

1. Name two approaches used to design a solution.

Answer: Procedural and object-oriented

**Evolution of C# and .NET**

**Programming Languages**

Programming began in the 1940s, when programmers toggled switches on the front of computers to enter programs and data into memory. In the 1950s, assembly languages replaced the binary notation by using mnemonic symbols to represent the instructions for the computer. Assembly languages are low-level programming languages. High-level languages came into existence in the late 1950s with FORTRAN and COBOL. High-level languages are designed to be accessible to humans—easy to read and write and close to the English language.

Dennis Ritchie is credited with developing the C language. C++ is an extension of C; Bjarne Stroustrup at Bell Labs is considered the father of C++ for his design work in the early 1980s. Visual Basic, introduced in 1991, is derived from BASIC, a language developed in the 1960s. The earlier versions of Visual Basic did not facilitate development using an object-oriented approach. Earlier versions of Visual Basic did, however, facilitate easy creation of Windows-based graphical user interfaces (GUIs). Java was introduced in 1995 and was originally called Oak.

C# is one of the newer programming languages. It conforms closely to C and C++, but many developers consider it akin to Java. There are a number of similarities between the languages. It has the rapid graphical user interface (GUI) features of previous versions of Visual Basic, the added power of C++, and object-oriented class libraries similar to Java. C# was designed from scratch to work with the new programming paradigm,

**.NET**

NET is an environment in which programs run. It is not an operating system, but rather a layer between the operating system and other applications. As such, it provides a platform for developing and running code that is easy to use. Included in Visual Studio are tools for typing program statements, and compiling, executing, and debugging applications. Included as part of .NET are multi-language independence capability, Framework class library with over 2500 reusable types (classes), and the capability of creating dynamic web pages, web services and scalable components.

**Quick Quiz**

1. Give three examples of high-level programming languages.

Answer: C#, Java, Visual Basic

1. True or False: .NET has been around since the mid-1990s.

Answer: False

1. True or False: Assembly is an example of a low-level programming language.

Answer: True

1. Identify two new concepts included as part of .NET.

Answer: Use of multiple languages to write the solution for one application and an extensive class library

**Why C#?**

Compilers targeting the .NET platform are available for a variety of programming languages.

* C# was *the* language created for .NET and was designed from scratch to work with .NET.
* Most of the .NET Framework classes were written using the C# programming language.
* C#, in conjunction with the .NET Framework classes, offers an exciting vehicle to incorporate and use emerging Web standards.
* C# is a simple, object-oriented language.
* Using the Visual Studio IDE and the .NET Framework, C# provides an easy way to create graphical user interfaces.
* On December 13, 2001, the European Computer Manufacturers Association (ECMA) General Assembly ratified C# and its common language infrastructure (CLI) specifications into international standards. C# is being ported to other platforms such as Linux.
* C# represents the next generation of languages.

**Quick Quiz**

1. \_\_\_\_\_\_\_\_\_\_\_\_\_ was the language created for .NET.

Answer: C#

1. True or False: C# can be used for creating Web applications, but not Windows applications.

Answer: False

1. True or False: All of the .NET Framework classes were written in C#.

Answer: False

1. Identify two reasons to use C# for developing applications.

Answer: C# is a simple object-oriented language. You can create graphical user interfaces, web application or scalable components.

**Types of Applications Developed with C#**

**Web Applications**

Programmers can use C# to quickly build applications that run on the Web for end users to view through browser-neutral user interfaces (UIs). **ASP.NET** is a programming framework that lets you create these types of applications.

**Windows Applications**

Designed for desktop use and for a single platform, they run on PC desktops much like your favorite word-processing program. Using the Integrated Development Environment (IDE) of Visual Studio, you can drag and drop controls such as buttons, text boxes, and labels onto an application.

**Console Applications**

Requests to the operating system are sent to display text on the command console display or to retrieve data from the keyboard. These are the simplest types of applications to create.

In addition to these applications, class libraries and stand-alone components (.dlls), smart device applications, and services can also be created using C#.

**Quick Quiz**

1. True or False: Output is displayed on the command windows for console applications.  
   Answer: True
2. True or False: ASP.NET is used to create Windows applications.

Answer: False

1. List two types of controls that can be added to a Windows application.   
   Answer: Button, label
2. IDE stands for \_\_\_\_\_\_\_\_\_\_\_\_\_.

Answer: Integrated Development Environment

**Exploring the First C# Program**

Readability is important. As far as the compiler is concerned, you could actually type the entire program without touching the Enter key. Use a consistent style when you develop your programs. Curly braces { } could be matched and appear on separate lines, by themselves. Another convention is to place the opening curly brace on the same line as the heading and align the ending curly brace with the first character of the heading. Whichever style you choose, be consistent.

**Quick Quiz**

1. True or False: No more than 256 characters can be placed on a single line when you are typing your program statements.  
   Answer: False
2. True or False: You must type line numbers before you type the program statement.  
   Answer: False
3. You should use a(n) \_\_\_\_\_\_\_\_\_\_\_\_\_ style when you type your program statements.  
   Answer: consistent
4. List two things you might do to make your program more readable.   
   Answer: Group and indent statements; properly line up curly braces

**Elements of a C# Program**

There are a number of entries that appear in every program.

**Comments**

Comments are not considered instructions to the computer and therefore have no effect on the running of the program. When the program is compiled, comments are not checked for rule violations. With C#, three types of commenting syntax can be added to a program: in-line, multi-line, and XML document comments. Visual Studio displays comments in green, as does the textbook.

**In-Line Comments**

Use two forward slashes (//) to mark the rest of the line as a comment. It is considered a one-line comment because everything to the right of the forward slashes is ignored by the compiler.

**Multi-Line Comments**

Also called block comments, they are marked by typing a forward slash followed by an asterisk. You must end the comment using the opposite pattern of an asterisk followed by a forward slash. The comment can span many lines, and everything that appears between the comment symbols is treated as a comment.

**Using Directive**

Adding a using directive permits the use of classes found in a namespacewithout having to qualify their references using the namespace identifier. This can reduce the amount of typing that would be necessary without the directive. The most referenced namespace is System. The directive is added at the top of the program as shown below:

using System;

The System namespace contains classes that define commonly used types or classes such as the Console class. The Console class enables programmers to write to and read from the console window or keyboard.

**Namespace**

Use the keyword namespace to group semantically related types (classes) under a single umbrella. The body for the namespacemust be enclosed in curly braces ({ }). The namespace surrounds the class definition.

**Class Definition**

Everything in C# is designed around a class. Every class is named, and it is tradition to name the file the same name as the class name, except the filename will have a .cs extension. Like namespaces, each class definition must be enclosed in curly braces { }.

**Main ( ) Method**

The Main ( ) method can be placed anywhere inside the class. It is the “entry point” for all applications—where the program begins execution. A methodis a collection of one or more statements combined to perform an action. Typically, a method calls another method and can return a value to the calling method. Methods communicate with each other by sending arguments inside parentheses or as return values.

**Method Body—Statements**

Methods always appear with parentheses ( ). A call to any method always includes a set of parentheses following the method name identifier, as do signatures for methods. Statements that appear in a method are executed in sequential order. Once the end curly brace is encountered, control returns back to the calling method. When the closing curly brace is encountered with the Main ( ) method, the entire program ends.

WriteLine ( ) is defined in the Console classand can be called with or without arguments. To have a blank line displayed on the standard output device, type Console.WriteLine( ); without any arguments. Methods in this classinclude Read ( ), WriteLine ( ), and Write ( ). The method Write ( ) differs from WriteLine ( ) in that it does not automatically advance the carriage return to the next line when it finishes. Review Table 1-1. It describes the special escape sequence characters that can be added to the string literal included as an argument to the Write ( ) or WriteLine ( ) method.

The ReadKey( ) or Read ( ) methods are often used in a C# program to keep the output screen displayed until the user clicks a key on the keyboard. ReadLine ( ) allows multiple characters to be entered. It accepts characters until the Enter key is pressed.

**Quick Quiz**

1. What are the most common types of comments included in program? How are they distinguished?   
   Answer: Block or multi-line and in-line or one-line comments. Block comments use /\* and \*/ to end the comment. In-line comments are added using two forward slashes.
2. True or False: The Main ( ) method must be placed at the beginning of a program. .  
   Answer: False
3. True or False: The ReadLine ( ) method is terminated by typing end.  
   Answer: False
4. What is the relationship between the name of the file that contains the program statement and the name of the class?  
   Answer: They are named the same, except the file ends with a .cs file extension.

**Compiling, Building, and Running an Application**

To see the results of a program, you must type the statements, or source code, into a file, compile that code, and then execute the application.

**Typing Your Program Statements**

You have a couple of options. One approach is to type the source code statements using a simple text editor (such as Notepad). This offers the advantage of not requiring significant system resources. A second approach is to use the Visual Studio integrated development environment (IDE). The IDE is an interactive environment that enables you to type the source code, compile, and execute without leaving the IDE program

**Compilation and Execution Process**

The compilerchecks to make sure there are no rule violations in the source code (program statements). Once the code is successfully compiled, the compiler generates a file that ends with an .exe extension. This code is still not targeted to any specific CPU platform. A second required step, the just-in-timecompiler (JITer),reads the IL code and translates or produces the machine code that runs on the particular platform. After the code is translated in this second step, results can be seen.

**Compiling the Source Code Using Visual Studio IDE**

Use the built-in editor available with the Visual Studio IDE to type your program statement. You then compile the source code from one of the pull-down menu options in the IDE and execute the application using another menu option.

Use the built-in editor available with the Visual Studio IDE to type your program statements. You then compile the source code from one of the pull-down menu options in the IDE and execute the application using another menu option. As the Project Type, select **Console Application** for the Template. The Visual Studio IDE generates much of the code for you. You will want to change the name of the class and the source code file because Visual Studio names every class Class1. You can do this using the Use the **Solution Explorer** Window.

To compile the HelloWorldProgram project, select the **BUILD HelloWorldProgram** option on the **BUILD** Menu. To run the application, you can click **Start** or **Start Without Debugging** on the **DEBUG** menu bar. The software development cycle is iterative, and it is sometimes necessary to return to previous phases. After you type the additional statement, recompile and reexecute when there are no more syntax errors.

If you run your program using **DEBUG**>**Start Without Debugging**, instead of **DEBUG**> **Start**, you will not have to add the additional Console.Read ( ); or Console.ReadKey( ); to hold the screen. This is done automatically for you.

**Quick Quiz**

1. True or False: The output produced by the compiler is a run of your application.

Answer: False

1. Describe the process the program statements go through after being typed into an editor.

Answer: The statements are compiled. When no rule violations occur, the compiler converts the source code statements into Intermediate Language code. From there, a second step, just-in-time compilation, is necessary to convert the IL code to the native platform-specific code. Once in native code, the program can be executed and results seen.

1. True or False: Everything from typing the program statements to compiling and seeing the results of the run can be done within Visual Studio.

Answer: True

1. What additional program statement must be added to hold the screen for viewing if the application is run by selecting Debug> Start?

Answer: A Read ( ) or ReadKey( ) method must be called to hold the output screen.

**Debugging an Application**

The major categories of errors are syntax and run-time.

**Syntax Errors**

These are errors caught by the compiler and include things like typing errors created by misspelling a name or forgetting to end a statement with a semicolon. In C#, a single typing error may generate several error messages. The errors are displayed in the Task Listwindow found at the bottom of the IDE. A good exercise is to omit curly braces, semicolons, and misspelled words so that you can see what kind of error messages each mistake generates. You will then be more equipped to find those errors quickly.

It is best to fix the first error and then recompile rather than trying to fix all the errors in one pass because one error may cause several error messages to be displayed.

**Run-time Errors**

Run-time errors are more difficult to detect than syntax errors. A program containing run-time errors may compile without any problems, run, and produce results.

**Quick Quiz**

1. True or False: Syntax errors are more difficult to detect than run-time errors.

Answer: False

1. True or False: When you see the list of syntax errors, be sure to correct all of them prior to building the application a second time.

Answer: False

1. Misspelling a name or forgetting to end a statement with a semicolon are examples of runtime errors.

Answer: False

1. A single typing error can generate several error messages.

Answer: True

**PROGRAMMING EXAMPLE: PROGRAMMINGMESSAGE**

This example displays a message on the console screen. It begins by showing a problem specification that details the problem definition. Focus is placed on understanding the problem definition.

No input is needed. The output produced is "Programming can be FUN!”. A prototype for the output is shown. An algorithm is developed using a flowchart. Once the design is completed and it is verified that the algorithm produces the correct output, the algorithm is translated into source code. You can type source code statements into the computer using Visual Studio IDE. From the generated code, you can remove the XML-style comments. The signature for Main ( ) can also be modified by removing the arguments “string[ ] args” inside the parentheses so that it has an empty argument list. Also change the name of the class to a name that better represents what the application is doing.

Note that the statements inside the Main ( ) method are executed in sequential order. The   
ReadKey ( ) method is executed after the two WriteLine ( ) methods. ReadKey ( ) is used in a program to keep the output screen displayed until the user clicks a key. During implementation, the source code is compiled to check for rule violations.

Just because you have no compiler syntax errors and receive output does not mean the results are correct. During this final step, test the program and ensure you have the correct result. The output should match your prototype.

**Quick Quiz**

1. What is a prototype?

Answer: Mock-up of the desired end result

1. True or False: It is okay to remove [STAThread] from the program listing.

Answer: True

1. True or False: One of the first things you should do when you begin an application is change the name of the class and source code filename.

Answer: True

1. What should happen during the implementation phase of a project?

Answer: Source code is compiled to check for rule violations.

**Coding Standards**

Following coding standards when you design classes leads to better solutions and reduces the amount of time needed when you make changes to your program statements. Developing standards that you consistently adhere to increases coding efficiency.

**Pseudocode**

Use verbs to indicate what type of actions should be performed.

Group items and add indentation

**Resources**

Current C# Language Specifications –

<http://www.microsoft.com/en-us/download/details.aspx?id=7029>

Visual C# Express download –

<http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-csharp-express>

History of computing project –

<http://www.thocp.net/>

Intel processor information –

<http://www.intel.com>

Pascaline –

<http://www.thocp.net/hardware/pascaline.htm>

The Microsoft .NET Web site –

<http://www.microsoft.com/net>

The MSDN Visual C# home page –

<http://msdn2.microsoft.com/en-us/vcsharp/default.aspx>

U.S. Census Data on Computer and Internet Use –

<http://www.census.gov/cps/>

Wikipedia Free Encyclopedia –

<http://wikipedia.org>

Mono cross platform open source .NET framework –

<http://www.mono-project.com>

Microsoft Developer Network –

<http://msdn.microsoft.com/en-us/>

##### Discussion Questions

Some interesting topics of discussion in this chapter include:

* What types of applications do you think will characterize the sixth generation of computing or computing in the future?
* What characteristics make up a class representing a city?
* What other development environments other than Visual Studio are currently available to be used with C#?
* What are the ways you can run a C# program?
* How do syntax errors differ from logic errors?

**Projects to Assign**

All of the Multiple Choice Exercises, Problems 1-20

Odd-numbered Short Answer Exercises, Problems 21-27

Programming Exercises, Problems 1, 2, 8, 9, and 10

**Key Terms**

* **ASP.NET**: a Microsoft programming framework that lets you create applications that run on a Web server and delivers functionality through a browser
* **abstracting out the attributes**: determining the data characteristics associated with a class object
* **algorithm**: a clear, unambiguous, step-by-step process for solving a problem
* **American Standard Code for Information Interchange (ASCII)**: character set that consists of the alphabet for the English language, plus numbers and symbols; the first 128 characters correspond to the Unicode character set
* **application software**: programs developed to perform a specific task
* **assemblers**: programs that convert the assembly programming language into native code
* **behaviors**:processes on the data of a class
* **beta version**:working version that has not been fully tested and may still contain errors
* **block comment**: a comment that can span more than one line that is marked by a forward slash followed by an asterisk (/\*) and ended with the opposite pattern (\*/)
* **bugs**:program errors
* **bytecode**:Java’s intermediate language
* **C# (C sharp)**: One of the newer high-level programming languages that is part of the .NET paradigm
* **characteristics**:attributes of a class
* **class**:logical groupings of data and behavior members
* **class diagram**:one of the primary modeling tools used by object-oriented programmers
* **COBOL (Common Business Oriented Language)**: a high-level programming language
* **Common Language Runtime (CLR)**:execution engine of .NET
* **compiler**: program used to check to make sure there are no rule violations in the program statement and then translates the instructions written in a high-level programming language into machine-readable format
* **console applications**: display text on the command console display
* **constant**: a data item that should keep the same value throughout the program
* **definition of the method**: method heading along with the complete body of the method
* **desk check**:mimicking the computer by walking through the program steps
* **divide and conquer**:break the problem into subtasks
* **domain**:range of the values for each input item
* **encapsulated**: packaging of the characteristics and behaviors together to form a class
* **entity**:a person, place, or thing
* **escape character**: a special string combination that begins with the backslash; when used in combination with other characters, it has special meaning
* **FORTRAN** **(Formula Translator)**: a high-level programming language
* **Framework Base Classes**: a collection of over 2,500 reusable types (classes) included as part of the .NET Framework
* **general-purpose computer**:systems are electronic devices that process data and are composed of hardware and software
* **hardware**:physical devices that you can touch
* **high-level languages**:instructions written in English-like statements
* **identifier**: the name given to a user-supplied or predefined entry
* **IL**: when there are no rule violations, the compiler converts the source code into intermediate language; also called MSIL
* **information hiding**:making the data member private and accessible to that class only
* **inheritance**: way to define subclasses of data objects that share some or all of the parent’s class characteristics
* **in-line comment**: one-line comment that is marked by two forward slashes (//) and ends when the enter key is pressed
* **instance**:one example of the class
* **instantiate**: create an instance of the class
* **instruction cycle**:consists of fetching a program instruction from memory, decoding the instruction, executing it, and then storing the result in memory
* **Intermediate Language (IL)**: all languages targeting the .NET platform compile into this intermediate language
* **interpreters**:check for rule violations line by line
* **iterative approach**:an approach that involvesgoing back to the analysis or design stage to make modification when problems are identified
* **Java**:a high-level programming language introduced in 1995 that was originally called Oak
* **JITer**:program that reads the intermediate language (IL) code and translates or produces the machine code that runs on the particular platform; after the code is translated in this second step, results can be seen
* **just-in-time (JIT) compilation**: second step that is required before you see the results of the application; it converts the IL code to the platform’s native code
* **keywords**: reserved words that have special predefined meanings
* **logic errors**: errors that cause an abnormal termination of the program or just produce incorrect results
* **low-level programming languages**: programming languages that are not considered close to the English language in terms of readability
* **main memory**:a device that holds instructions and data; also called primary storage or random access memory
* **method**: collection of one or more statements combined to perform an action
* **method invocation**: call to a method
* **methodologies**: approaches used to solve computer-related problems
* **Microsoft Intermediate Language (MSIL)**: when there are no rule violations, the compiler converts the source code into intermediate language; also called IL
* **multi-language independence**: development of an application using a number of programming languages
* **multi-line comment**: a comment that can span more than one line that is marked by a forward slash followed by an asterisk (/\*) and ended with the opposite pattern (\*/)
* **namespace**: groups semantically related types under a single umbrella
* **native code**:machine language code of a particular computer
* **.NET**:an environment in which programs run that was designed to be a new programming paradigm
* **object**: instance of the class
* **object-oriented analysis**, **design**, **and programming**:focus is on determining the objects you want to manipulate rather than the processes or logic required to manipulate the data
* **object-oriented approach**: the focus is on determining the data characteristics and the methods or behaviors that operate on the data
* **operating systems**:types of programs that oversee and coordinate the resources on the machine
* **procedural programming**:process-oriented approach that focuses on the processes that data undergoes from input until meaningful output is produced
* **programming language**: used to write instructions for the computer
* **programs**: sets of instructions telling the computer exactly what to do
* **prototype**:mock-up of screens depicting the look of the final output
* **pseudocode**: steps are written in “pseudo” or approximate code format, which looks like English statements
* **random-access memory (RAM)**: a device that holds instructions and data; also called main memory or primary storage
* **redistributable version**: smaller download than the Software Development Kit and includes the CLR and class libraries
* **run-time error**:form of logic error; run-time errors normally cause program crashes (stopping execution) and the reporting of error messages
* **scalable component development**: segments of code, created as separate entities, that can be stored independently, combined, and reused in many applications
* **semantic meaning**: specific way in which a program statement or keyword should be used
* **signature**: the name of the method and its argument list
* **software**: programs; the sets of instructions that make the hardware function
* **software maintenance**: upgrade or change an application
* **source code**: program statements written using a programming language
* **specifications**: describes what the program should accomplish
* **stepwise refinement**:refine the logic by dividing and conquering
* **syntax**: the set of rules of the language that must be followed
* **test plans**:plans for how the program will be tested to ensure its correctness
* **testing**: the process of verifying the correctness of a program to ensure that you get consistently accurate results
* **top-down design**:way to conquer each of the subtasks by further decomposing them
* **Visual Studio**: An IDE that includes a suite of products including several programming languages, along with a large collection of development and debugging tools
* **Web Forms**: ASP.NET programmable Web pages that serve as a UI for Web applications
* **Windows applications**: applications designed for desktop single platform use
* **XML (Extensible Markup Language)**: language that provides a format for describing data using tags similar to HTML tags
* **XML documentation comments**: comments that can be used to generate XML documentation file