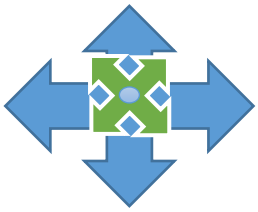
>>>>>>>>>>>>>>>>>>>Introduction To Computers, Programs And Java<<<<<<<<<<<<<<<<<<

Brought To You By



{My Life} Programming School

Learn Java By Reading First, Then Watch Someone Tackling Exercises Programmatically By Visiting The Link.
https://www.mylifeprogrammingschool.com

1.1 Introduction

{Key Point} The central theme of these slides is to learn how to solve problems by writing a program. So, what is programming? The term *programming* means to create (or develop) software, which is also called a *program.* In basic terms, software contains the instructions that tell a computer—or a computerized device—what to do. Software is all around you, even in devices that you might not think would need it. Of course, you expect to find and use software on a personal computer, but software also plays a role in running airplanes, cars, cell phones, and even toasters.

Software developers create software with the help of powerful tools called *programming languages.* These slides teach you how to create programs by using the Java programming language. These slides teach you how to create programs by using the Java programming language.

1.2 What Is a Computer?

{Key Point} A computer is an electronic device that stores and processes data. A computer includes both *hardware* and *software.* In general, hardware comprises the visible, physical elements of the computer, and software provides the invisible instructions that control the hardware and make it perform specific tasks. Knowing computer hardware isn't essential to learning a programming language, but it can help you better understand the effects that a program's instructions have on the computer and its components. A computer consists of the following major hardware components

- A central processing unit (CPU)
- Memory (main memory)
- Storage devices (such as disks and CDs)
- Input devices (such as the mouse and keyboard)
- Output devices (such as monitors and printers)
- Communication devices (such as modems and network interface cards)

A computer's components are interconnected by a subsystem called a *bus.* You can think of a bus as a sort of system of roads running among the computer's components; data and power travel along the bus from one part of the computer to another. In personal computers, the bus is built into the computer's *motherboard*, which is a circuit case that connects all of the parts of a computer together.

1.2.1 Central Processing Unit

The *central processing unit (CPU)* is the computer's brain. It retrieves instructions from memory and executes them. The CPU usually has two components: a *control unit* and an *arithmetic/logic unit.* The control unit controls and coordinates the actions of the other components. The arithmetic/logic unit performs numeric operations (addition, subtraction, multiplication, division) and logical operations (comparisons). Today's CPUs are built on small silicon semiconductor chips that contain millions of tiny electric switches, called *transistors*, for processing information.

Every computer has an internal clock, which emits electronic pulses at a constant rate. These pulses are used to control and synchronize the pace of operations. A higher clock *speed* enables more instructions to be executed in a given period of time. The unit of measurement of clock speed is the *hertz* (*Hz*), with 1 hertz equaling 1 pulse per second. In the 1990s, computers measured clocked speed in *megahertz* (*MHz*), but CPU speed has been improving continuously; the clock speed of a computer is now usually stated in *gigahertz (GHz)*. Intel's newest processors run at about 3 GHz. CPUs were originally developed with only one core. The *core* is the part of the processor that performs the reading and executing of instructions. In order to increase CPU processing power, chip manufacturers are now producing CPUs that contain multiple cores. A multicore CPU is a single component with two or more independent cores. Today's consumer computers typically have two, three, and even four separate cores. Soon, CPUs with dozens or even hundreds of cores will be affordable.

1.2.2 Bits and Bytes

Before we discuss memory, let's look at how information (data and programs) are stored in a computer. A computer is really nothing more than a series of switches. Each switch exists in two states: on or off. Storing information in a computer is simply a matter of setting a sequence of switches on or off. If the switch is on, its value is 1. If the switch is off, its value is 0.

These 0s and 1s are interpreted as digits in the binary number system and are called *bits* (binary digits). The minimum storage unit in a computer is a *byte.* A byte is composed of eight bits. A small number such as **3** can be stored as a single byte. To store a number that cannot fit into a single byte, the computer uses several bytes. Data of various kinds, such as numbers and characters, are encoded as a series of bytes.

As a programmer, you don't need to worry about the encoding and decoding of data, which the computer system performs automatically, based on the encoding scheme. An *encoding scheme* is a set of rules that govern how a computer translates

characters, numbers, and symbols into data the computer can actually work with. Most schemes translate each character into a predetermined string of bits. In the popular ASCII encoding scheme, for example, the character **C** is represented as **01000011** in one byte.

A computer's storage capacity is measured in bytes and multiples of the byte, as follows:

- A *kilobyte (KB)* is about 1,000 bytes.
- A *megabyte (MB)* is about 1 million bytes.
- A *gigabyte (GB)* is about 1 billion bytes.
- A *terabyte (TB)* is about 1 trillion bytes.

A typical one-page word document might take 20 KB. Therefore, 1 MB can store 50 pages of documents and 1 GB can store 50,000 pages of documents. A typical two-hour high resolution movie might take 8 GB, so it would require 160 GB to store 20 movies.

1.2.3 Memory

A computer's *memory* consists of an ordered sequence of bytes for storing programs as well as data that the program is working with. You can think of memory as the computer's work area for executing a program. A program and its data must be moved into the computer's memory before they can be executed by the CPU. Every byte in the memory has a *unique address*. The address is used to locate the byte for storing and retrieving the data. Since the bytes in the memory can be accessed in any order, the memory is also referred to as *random-access memory (RAM)*. The word "Create" can be store in memory as follows.

Memory Address	Memory content	Encoded letter
.		
.		
.		
3000	01000011	Encoding for character 'C'
3001	01110010	Encoding for character 'r'
3002	01100101	Encoding for character 'e'
.		
.		
.		

Memory stores data and program instructions in uniquely addressed memory locations. Generally speaking, the more RAM a computer has, the faster it can operate, but there are limits to this simple rule of thumb. A memory byte is never empty, but its initial content may be meaningless to your program. The current content of a memory byte is lost whenever new information is placed in it.

1.2.4 Storage Devices

A computer's memory (RAM) is a volatile form of data storage: any information that has been stored in memory (i.e., saved) is lost when the system's power is turned off. Programs and data are permanently stored on *storage devices* and are moved, when the computer actually uses them, to memory, which operates at much faster speeds than permanent storage devices can.

There are three main types of storage devices:

- Magnetic disk drives
- Optical disc drives (CD and DVD)
- USB flash drives

Drives are devices for operating a medium, such as disks and CDs. A storage medium physically stores data and program instructions. The drive reads data from the medium and writes data onto the medium. A computer usually has at least one hard disk drive. *Hard disks* are used for permanently storing data and programs.

CD stands for compact disc. There are two types of CD drives: CD-R and CD-RW. A *CD-R* is for read-only permanent storage; the user cannot modify its contents once they are recorded. A *CD-RW* can be used like a hard disk; that is, you can write data onto the disc, and then
overwrite that data with new data. A single CD can hold up to 700 MB.

DVD stands for digital versatile disc or digital video disc. DVDs and CDs look alike, and you can use either to store data. A DVD can hold more information than a CD; a standard DVD's storage capacity is 4.7 GB. Like CDs, there are two types of DVDs: DVD-R (readonly) and DVD-RW (rewritable).

Universal serial bus (USB) connectors allow the user to attach many kinds of peripheral devices to the computer. You can use a USB to connect a printer, digital camera, mouse, external hard disk drive, and other devices to the computer. A USB *flash drive* is a device for storing and transporting data.

1.2.5 Input and Output Devices

Input and output devices let the user communicate with the computer. The most common input devices are keyboards and mice. The most common output devices are monitors and printers.

Keyboard - A keyboard is a device for entering input. Compact keyboards are available without a numeric keypad.

Mouse - A *mouse* is a pointing device. It is used to move a graphical pointer (usually in the shape of an arrow) called a *cursor* around the screen or to click on-screen objects (such as a button) to trigger them to perform an action.

Monitor - The *monitor* displays information (text and graphics). The screen resolution and dot pitch determine the quality of the display.

The *screen resolution* specifies the number of pixels in horizontal and vertical dimensions of the display device. *Pixels* (short for “picture elements”) are tiny dots that form an image on the screen.

1.2.6 Communication Devices

Computers can be networked through communication devices, such as a dial-up modem (modulator/demodulator), a DSL or cable modem, a wired network interface card, or a wireless adapter.

■ A dial-up modem uses a phone line and can transfer data at a speed up to 56,000 bps (bits per second).
■ A digital subscriber line (DSL) connection also uses a standard phone line, but it can transfer data 20 times faster than a standard dial-up modem.
■ A cable modem uses the cable TV line maintained by the cable company and is generally faster than DSL.
■ A network interface card (NIC) is a device that connects a computer to a local area network (LAN). LANs are commonly used in universities, businesses, and government agencies. A high-speed NIC called 1000BaseT can transfer data at 1,000 million bits per second (mbps).
■ Wireless networking is now extremely popular in homes, businesses, and schools. Every laptop computer sold today is equipped with a wireless adapter that enables the computer to connect to a local area network and the Internet.

1.3 Programming Languages

{Key Point} Computer programs, known as software, are instructions that tell a computer what to do. Computers do not understand human languages, so programs must be written in a language a
computer can use. There are hundreds of programming languages, and they were developed to make the programming process easier for people. However, all programs must be converted into the instructions the computer can execute.

1.3.1 Machine Language

A computer’s native language, which differs among different types of computers, is its *machine language*—a set of built-in primitive instructions. These instructions are in the form of binary code, so if you want to give a computer an instruction in its native language, you have to enter the instruction as binary code. For example, to add two numbers, you might have to write an instruction in binary code, like this:

1101101010011010

1.3.2 Assemble Language

Programming in machine language is a tedious process. Moreover, programs written in machine language are very difficult to read and modify. For this reason, *assembly language* was created in the early days of computing as an alternative to machine languages. Assembly

language uses a short descriptive word, known as a *mnemonic*, to represent each of the machine-language instructions. For example, the mnemonic **add** typically means to add numbers and **sub** means to subtract numbers. To add the numbers **2** and **3** and get the result, you might write an instruction in assembly code like this:

```
add 2, 3, result
```

Assembly languages were developed to make programming easier. However, because the computer cannot execute assembly language, another program—called an *assembler*—is used to translate assembly-language programs into machine code. Writing in assembly requires that you know how the CPU works. Assembly language is referred to as a *low-level language*, because assembly language is close in nature to machine language and is machine dependent.

1.3.3 High-Level Language

In the 1950s, a new generation of programming languages known as *high-level languages* emerged. They are platform independent, which means that you can write a program in a high-level language and run it in different types of machines. High-level languages are English-like and easy to learn and use. The instructions in a high-level programming language are called *statements.* Here, for example, is a high-level language statement that computes the area of a circle with a radius of **5**:

```
area = 5 * 5 * 3.14159;
```

A program written in a high-level language is called a *source program* or *source code.* Because a computer cannot execute a source program, a source program must be translated into machine code for execution. The translation can be done using another programming tool called an *interpreter* or a *compiler.*

■ *An interpreter reads one statement from the source code, translates it to the machine code or virtual machine code, and then executes it right away. Note that a statement from the source code may be translated into several machine instructions.*
■ *A compiler translates the entire source code into a machine-code file, and the machine-code file is then executed.*

1.4 Operating System

{Key Point} The operating system (OS) is the most important program that runs on a computer. The OS manages and controls a computer's activities. The operating system (OS) is the most important program that runs on a computer. The OS manages and controls a computer's activities.

The major tasks of an operating system are as follows:

- Controlling and monitoring system activities

- Allocating and assigning system resources
- Scheduling operations

Operating systems perform basic tasks, such as recognizing input from the keyboard, sending output to the monitor, keeping track of files and folders on storage devices, and controlling peripheral devices, such as disk drives and printers. An operating system must also ensure that different programs and users working at the same time do not interfere with each other. In addition, the OS is responsible for security, ensuring that unauthorized users and programs are not allowed to access the system.

The operating system is responsible for determining what computer resources a program needs (such as CPU time, memory space, disks, input and output devices) and for allocating and assigning them to run the program.

The OS is responsible for scheduling programs' activities to make efficient use of system resources. Many of today's operating systems support techniques such as *multiprogramming*, *multithreading*, and *multiprocessing* to increase system performance.

Multiprogramming allows multiple programs to run simultaneously by sharing the same CPU. The CPU is much faster than the computer's other components. As a result, it is idle most of the time—for example, while waiting for data to be transferred from a disk or waiting for other system resources to respond. A multiprogramming OS takes advantage of this situation by allowing multiple programs to use the CPU when it would otherwise be idle.

Multithreading allows a single program to execute multiple tasks at the same time. For instance, a word-processing program allows users to simultaneously edit text and save it to a disk. In this example, editing and saving are two tasks within the same application. These two
tasks may run concurrently.

Multiprocessing, or *parallel processing*, uses two or more processors together to perform subtasks concurrently and then combine solutions of the subtasks to obtain a solution for the entire task. It is like a surgical operation where several doctors work together on one patient.

1.5 Java, the World Wide Web, and Beyond

{Key Point} Java is a powerful and versatile programming language for developing software running on mobile devices, desktop computers, and servers. Java syntax is defined in the Java language specification, and the Java library is defined in the Java API. The JDK is the software for developing and running Java programs. An IDE is an integrated development environment for rapidly developing programs.

1.6 A Simple Java Program

{Key Point} A Java program is executed from the **main** method in the class. Let's begin with a simple Java program that displays the message **Welcome to Java!** on the console. (The word *console* is an old computer term that refers to the text entry

and display device of a computer. *Console input* means to receive input from the keyboard, and *console output* means to display output on the monitor.) The program is shown in Listing 1.1.

LISTING 1.1 Welcome.java

```
1 public class Welcome {
2 public static void main(String[] args) {
3 // Display message Welcome to Java! on the console
4 System.out.println("Welcome to Java!");
5 }
6 }
```

```
Welcome to Java!
```

Note that the line numbers are for reference purposes only; they are not part of the program. So, don't type line numbers in your program.

Line 1 defines a class. Every Java program must have at least one class. Each class has a name. By convention, class names start with an uppercase letter. In this example, the class name is **Welcome**.

Line 2 defines the **main** method. The program is executed from the **main** method. A class may contain several methods. The **main** method is the entry point where the program begins execution. A method is a construct that contains statements. The **main** method in this program contains the **System.out.println** statement. This statement displays the string **Welcome to Java!** on the console (line 4). *String* is a programming term meaning a sequence of characters. A string must be enclosed in double quotation marks. Every statement in Java ends with a semicolon (**;**), known as the *statement terminator. Reserved words*, or *keywords*, have a specific meaning to the compiler and cannot be used for other purposes in the program. For example, when the compiler sees the word **class**, it understands that the word after **class** is the name for the class. Other reserved words in this program are **public**, **static**, and **void**.

Line 3 is a *comment* that documents what the program is and how it is constructed. Comments programmers to communicate and understand the program. They are not programming statements and thus are ignored by the compiler. In Java, comments are preceded by two slashes (***//***) on a line, called a *line comment,* or enclosed between ***/**** and ****/*** on one or several lines, called a *block comment* or *paragraph comment*.

When the compiler sees ***//***, it ignores all text after ***//*** on the same line. When it sees ***/****, it scans for the next ****/*** and ignores any text between ***/**** and ****/***. Here are examples of comments:

```
// This application program displays Welcome to Java!
/* This application program displays Welcome to Java! */
/* This application program
```

displays Welcome to Java! */

A pair of curly braces in a program forms a *block* that groups the program's components. In Java, each block begins with an opening brace (**{**) and ends with a closing brace (**}**). Every class has a *class block* that groups the data and methods of the class. Similarly, every method has a *method block* that groups the statements in the method. Blocks can be *nested*, meaning that one block can be placed within another, as shown in the following code.

{Tip} An opening brace must be matched by a closing brace. Whenever you type an opening brace, immediately type a closing brace to prevent the missing-brace error. Most Java IDEs automatically insert the closing brace for each opening brace. Java source programs are case sensitive. It would be wrong, for example, to replace **main** in the program with **Main**.

==========End Of Chapter One==========

Now You Can Checkout Chapter 1 Exercises And Their Corresponding Possible Solution On https://www.mylifeprogrammingschool.com/java