Objectives

- To understand computer basics, programs, and operating systems (§1.2–1.4).
- To describe the relationship between Java and the World Wide Web (§1.5).
- To understand the meaning of Java language specification, API, JDK, and IDE (§1.6).
- To write a simple Java program (§1.7).
- To display output on the console (§1.7).
- To explain the basic syntax of a Java program (§1.7).
- To create, compile, and run Java programs (§1.8).
- To use sound Java programming style and document programs properly (§1.9).
- To explain the differences between syntax errors, runtime errors, and logic errors (§1.10).
- To develop Java programs using NetBeans (§1.11).
- To develop Java programs using Eclipse (§1.12).

What is a Computer?

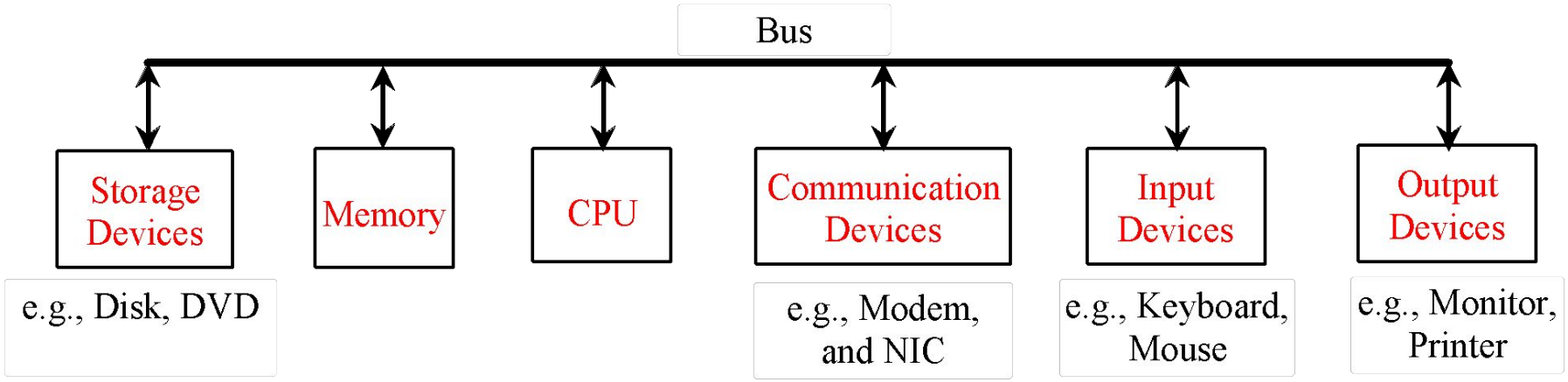
A **computer** is an electronic device that stores and processes data.

A computer includes both **hardware** and **software**.

Hardware is physical elements of the computer.

Software provides the invisible instructions that control the hardware.

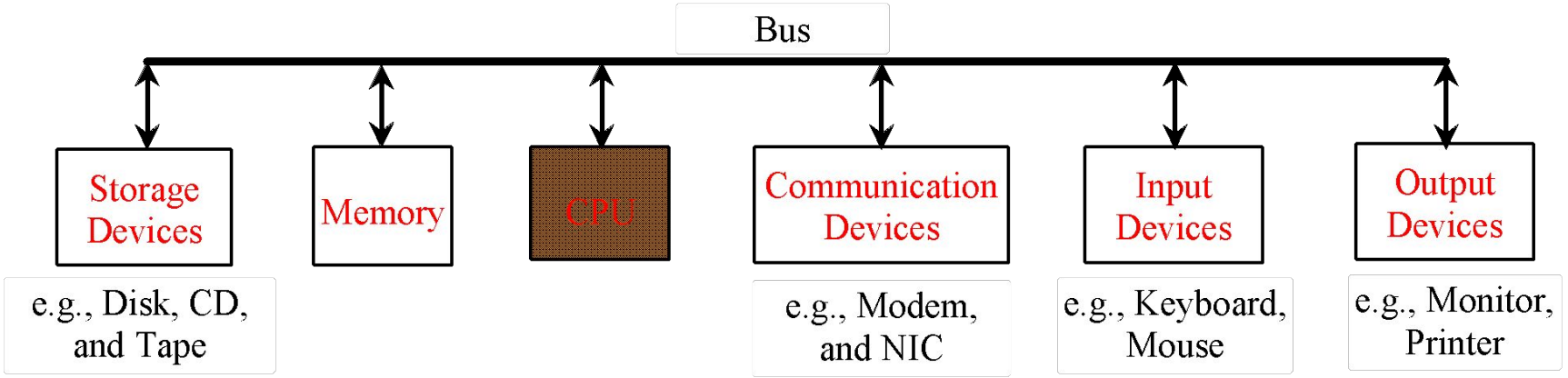
A computer's **major hardware elements** are CPU, memory, hard disk, monitor, printer, and communication devices.



CPU

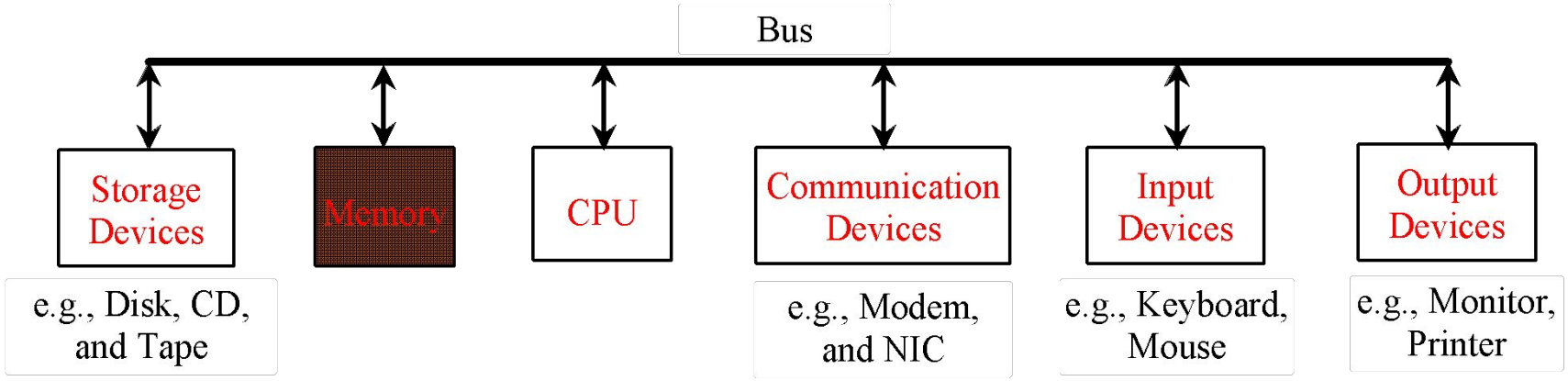
The central processing unit (CPU) is the brain of a computer. It retrieves(gets) instructions(directives) from memory and executes them. The CPU speed is measured in megahertz (MHz), with 1 megahertz equaling 1 million pulses per second. In every pulses one or more processes are performed.

The speed of the CPU has been improved continuously. If you buy a PC now, you can get an Intel Pentium 4 Processor at 3 gigahertz (1 gigahertz is 1000 megahertz).



Memory

Memory is to store data and program instructions for CPU to execute. **A memory unit** is an ordered sequence of bytes, each holds eight bits. A program and its data must be brought to memory before they can be executed. The current content of a memory byte is lost whenever new information is placed in it. Memory is volatile.



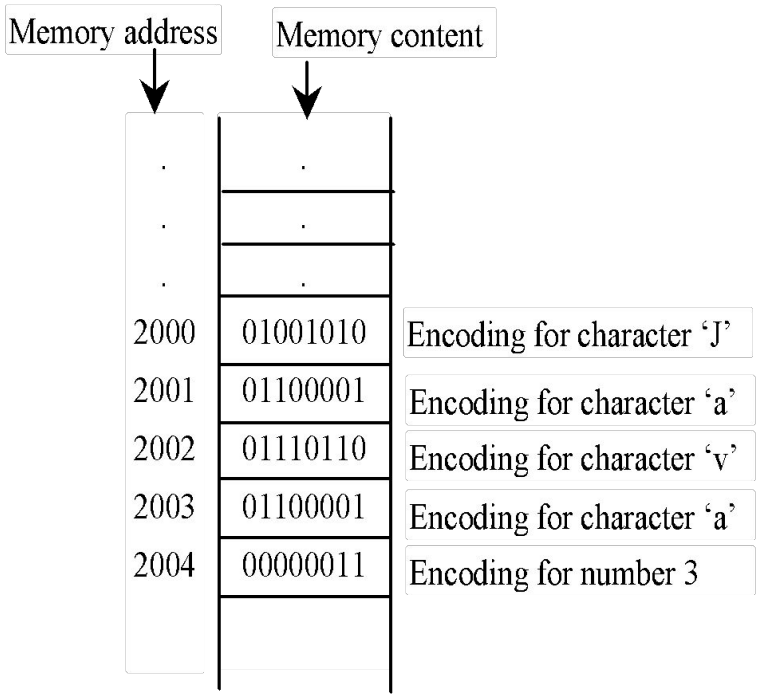
How Data is Stored?

Data of various kinds, such as numbers, characters, and strings, are encoded as a series of bits (0 and 1). A computer is a series of switches. Each switch exists in two states: on or off. If the switch is on, its value is 1. If the switch is off, its value is 0.

The programmers need not to be concerned about the encoding and decoding of data, which is performed automatically by the system based on the encoding scheme.

Some popular encoding schemes are ASCII, Unicode, UTF-8.

For example, character 'J' is represented by 01001010 in one byte. A small number such as 3 can be stored in a single byte. If computer needs to store a large number that cannot fit into a single byte, it uses a number of adjacent bytes. No two data can share or split a same byte. A byte is the minimum storage unit.



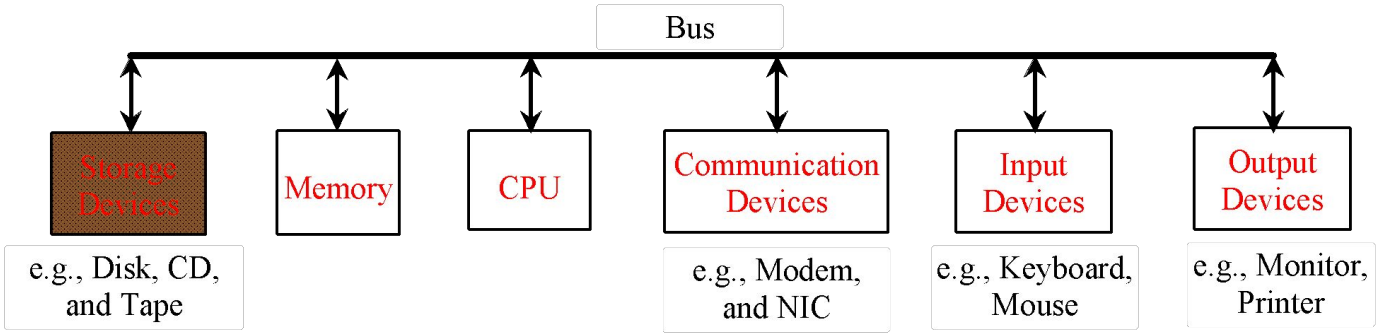
Storage Devices

Memory is volatile, because information is lost when the power is off. Programs and data are permanently stored on storage devices and are moved to memory when the computer actually uses them. There are three main types of storage devices: Disk drives (hard disks and floppy disks), CD drives (CD-R and CD-RW), and Tape drives.

Storage capacity is measured in bytes.

1 Kilobyte = 1024 bytes , 1 Megabyte = 1024 kilobytes

1 Gigabyte = 1024 megabytes, 1 Terabyte = 1024 gigabytes



Output Devices: Monitor

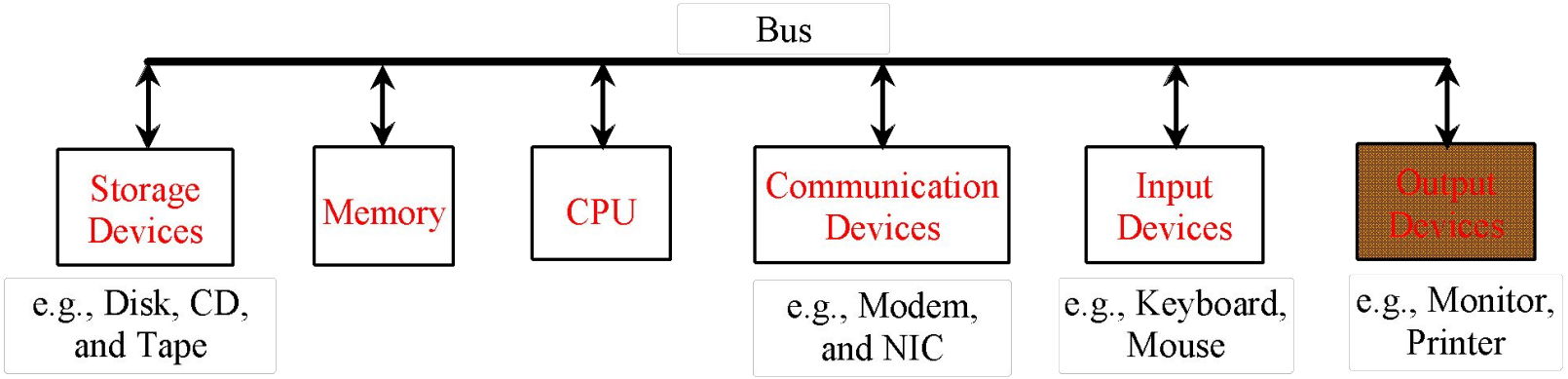
Monitor and printer are the most widely used output devices.

The monitor displays information (text and graphics). The 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.

The **dot pitch** is the amount of space between pixels, measured in millimeters.

The higher the resolution and the smaller the dot pitch, the sharper and clearer the image is.



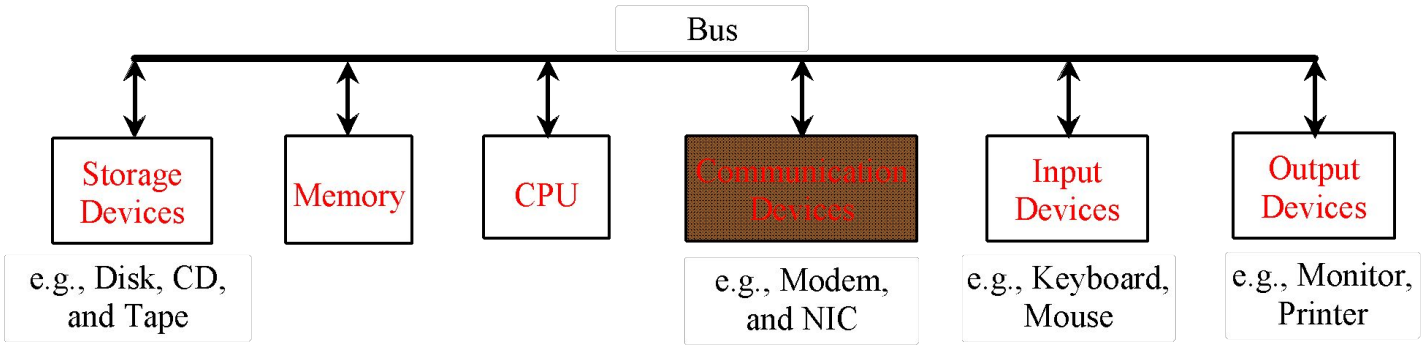
Communication Devices

A *regular modem* uses a phone line and can transfer data in a speed up to 56,000 bps (bits per second).

A *DSL* (digital subscriber line) also uses a phone line and can transfer data in a speed 20 times faster than a regular modem.

A *cable modem* uses the TV cable line maintained by the cable company. A cable modem is as fast as a DSL.

Network interface card (*NIC*) is a device to connect a computer to a local area network (LAN). The LAN is commonly used in business, universities, and government organizations. A typical type of NIC, called *100BaseT*, can transfer data at 100 mbps (million bits per second).



Programs

Computer *programs*, known as *software*, are instructions (directions) to the computer.

You tell a computer what to do through programs. Without programs, a computer is an empty machine. Computers do not understand human languages, so you need to use computer languages to communicate with them.

Programs are written using programming languages.

Programming Languages

Machine Language Assembly Language High-Level Language

Machine language is a set of primitive (basic) instructions built into every computer. The instructions are in the form of binary code, so you have to enter binary codes for various instructions.

Program with machine language is a process. Moreover the programs are highly difficult to read and modify. For example, to add two numbers, you might write an instruction in binary like this:

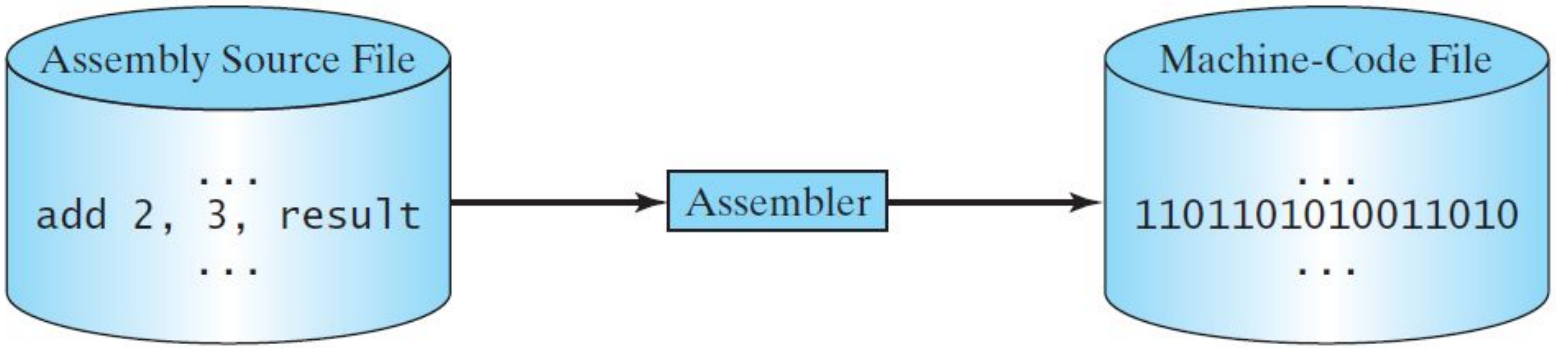
```
1101101010011010
```

Programming Languages

Machine Language Assembly Language High-Level Language

Assembly languages were developed to make programming easy. Since the computer cannot understand assembly language, however, a program called assembler is used to convert assembly language programs into machine code. For example, to add two numbers, you might write an instruction in assembly code like this:

ADDF3 R1, R2, R3



Programming Languages

Machine Language Assembly Language High-Level Language

The high-level languages are English-like and easy to learn and program. For example, the following is a high-level language statement that computes the area of a circle with radius 5:

```
area = 5 * 5 * 3.1415;
```

Popular High-Level Languages

Language	Description
Ada	Named for Ada Lovelace. The Ada language was developed for the US Department of Defense and is used mainly in defense projects.
BASIC	Beginner's All-purpose Symbolic Instruction Code. It was designed to be learned and used easily by beginners.
C	Developed at Bell Laboratories. C combines the power of an assembly language with the ease of use and portability of a high-level language.
C++	C++ is an object-oriented language, based on C.
C#	Pronounced "C Sharp." It is a hybrid of Java and C++ and was developed by Microsoft.
COBOL	COmmon Business Oriented Language. Used for business applications.
FORTRAN	FORmula TRANslation. Popular for scientific and mathematical applications.
Java	Developed by Sun Microsystems, now part of Oracle. It is widely used for developing platform-independent Internet applications.
Pascal	Named for Blaise Pascal. It is a simple, structured, general-purpose language primarily for teaching programming.
Python	A simple general-purpose scripting language good for writing short programs.
Visual Basic	Visual Basic was developed by Microsoft and it enables the programmers to rapidly develop graphical user interfaces.

Interpreting/Compiling Source Code

A program written in a high-level language is called a **source program** or **source code**.

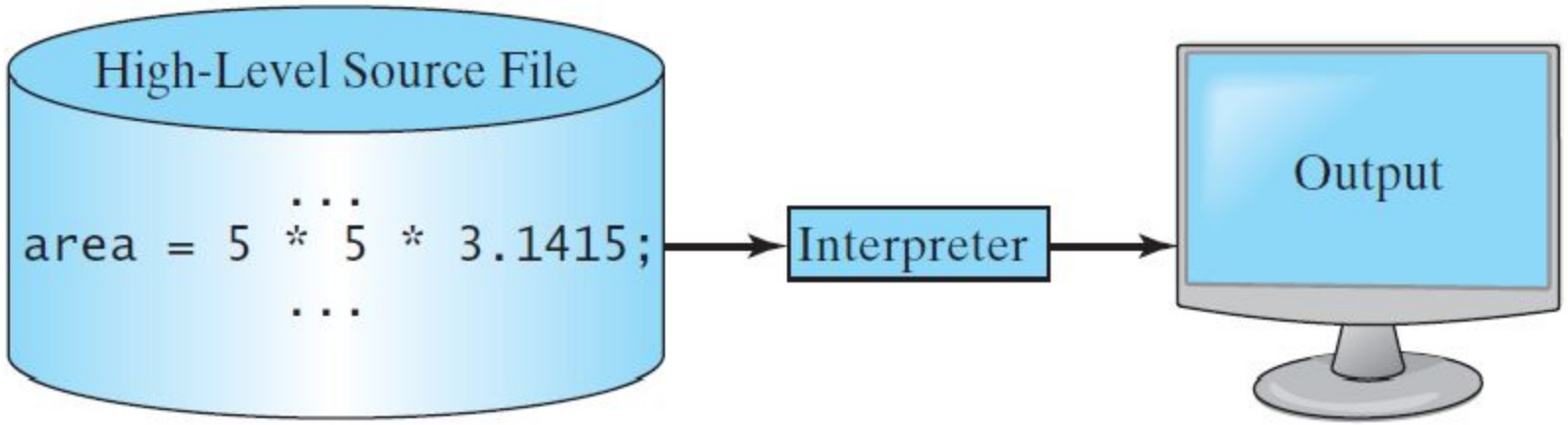
Because a computer cannot understand a source program, a source program must be translated into machine code for execution.

The translation can be done using another programming tool called an **interpreter** or a **compiler**.

Interpreting Source Code

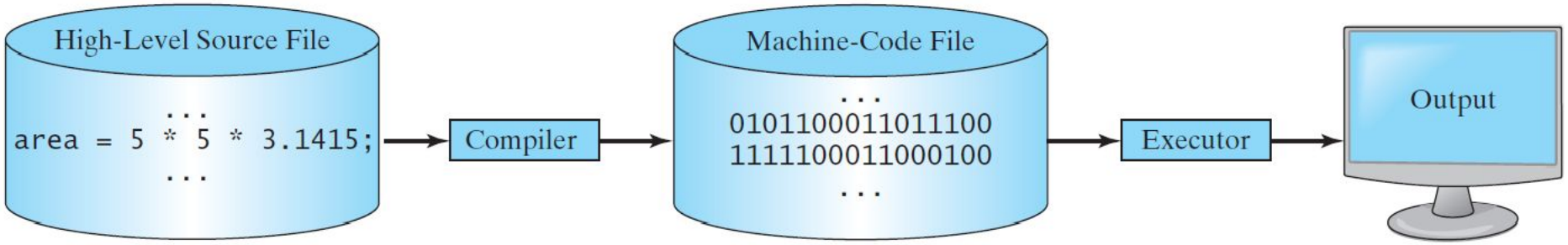
An interpreter reads one statement from the source code, translates it to the machine code, and then executes it right away(immediately), as shown in the following figure.

Note that a statement from the source code may be translated into several machine instructions.



Compiling Source Code

A compiler translates the entire source code into a machine-code file, and the machine-code file is then executed, as shown in the following figure.



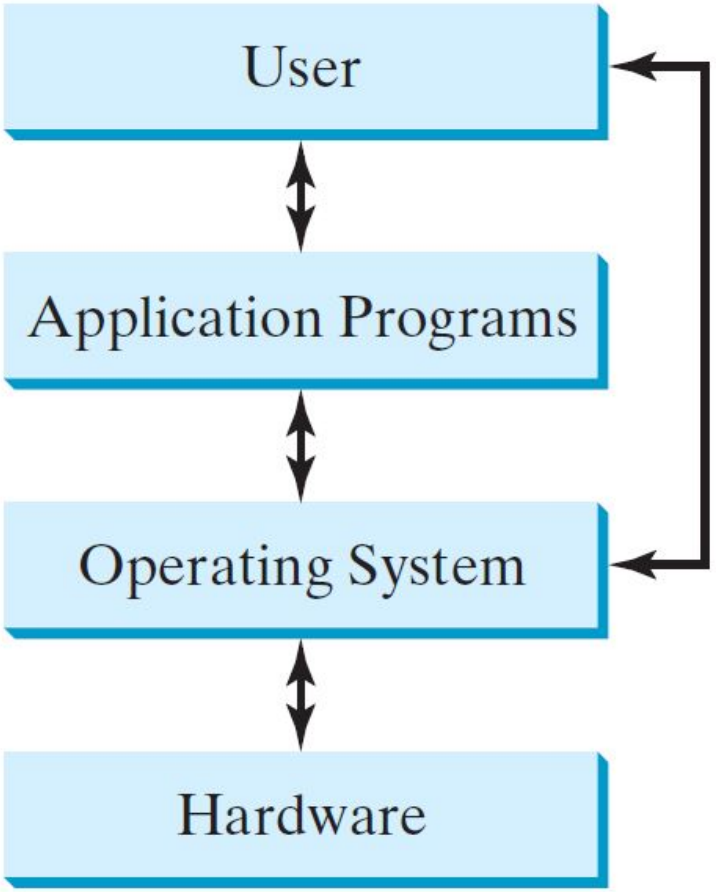
Operating Systems

The operating system (OS) is a program that manages and controls a computer's activities.

The popular operating systems for general-purpose computers are Microsoft Windows, Mac OS, and Linux.

Application programs, such as a Web browser or a word processor, cannot run unless an operating system is installed and running on the computer.

Users and applications access the computer's hardware via the operating system.



Why Java?

The answer is that Java enables users to develop and deploy applications on the Internet for servers, desktop computers, and small hand-held devices such as cell phone. The future of computing is being seriously influenced by the Internet, and Java promises to remain a big part of that future. Java is the Internet programming language.

- Java is a general purpose programming language.
- Java is the Internet programming language.

Java, Web, and Beyond

- Java can be used to develop standalone applications.
- Java can be used to develop applications running from a browser.
- Java can also be used to develop applications for hand-held devices.
- Java can be used to develop applications for Web servers.

Java's History

- James Gosling and Sun Microsystems
  - Developed by James Gosling's team at Sun Microsystems
- Oak
  - Name given the first version
- Java, May 20, 1995, Sun World
  - First introduced in Sun World Conferences
- HotJava
  - The first Java-enabled Web browser

JDK Versions

- JDK 1.02 (1995)
- JDK 1.1 (1996)
- JDK 1.2 (1998)
- JDK 1.3 (2000)
- JDK 1.4 (2002)
- JDK 1.5 (2004) is known as JDK 5 or Java 5
- JDK 1.6 (2006) is known as JDK 6 or Java 6
- JDK 1.7 (2011) is known as JDK 7 or Java 7
- JDK 1.8 (2014) is known as JDK 8 or Java 8

JDK Editions

- Java Standard Edition (J2SE)
  - J2SE can be used to develop client-side standalone applications or applets.
- Java Enterprise Edition (J2EE)
  - J2EE can be used to develop server-side applications such as Java servlets, Java ServerPages, and Java ServerFaces.
- Java Micro Edition (J2ME).
  - J2ME can be used to develop applications for mobile devices such as cell phones.

This book uses J2SE to introduce Java programming.

Popular Java IDEs

- NetBeans
- Eclipse

Installing Java and Eclipse

First install the Java, then Eclipse

- **Java SE Download**

  http://www.oracle.com/technetwork/java/javase/downloads/index.html

- **Eclipse Download**

http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/marsr

A Simple Java Program

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Animation

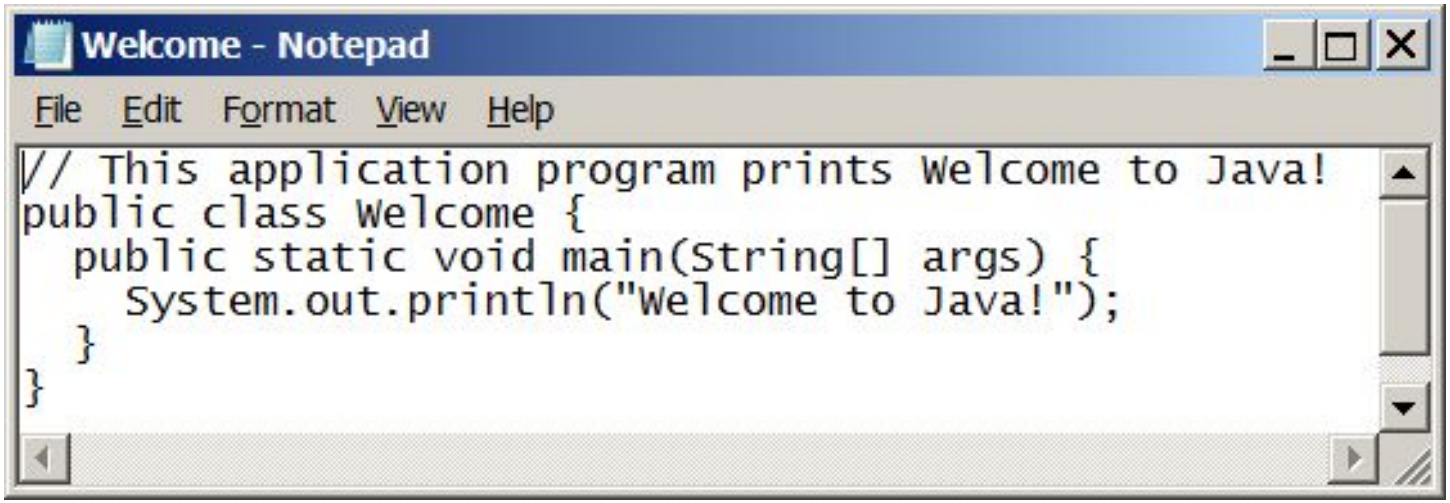
Welcome

Creating and Editing Using NotePad

To use NotePad, type

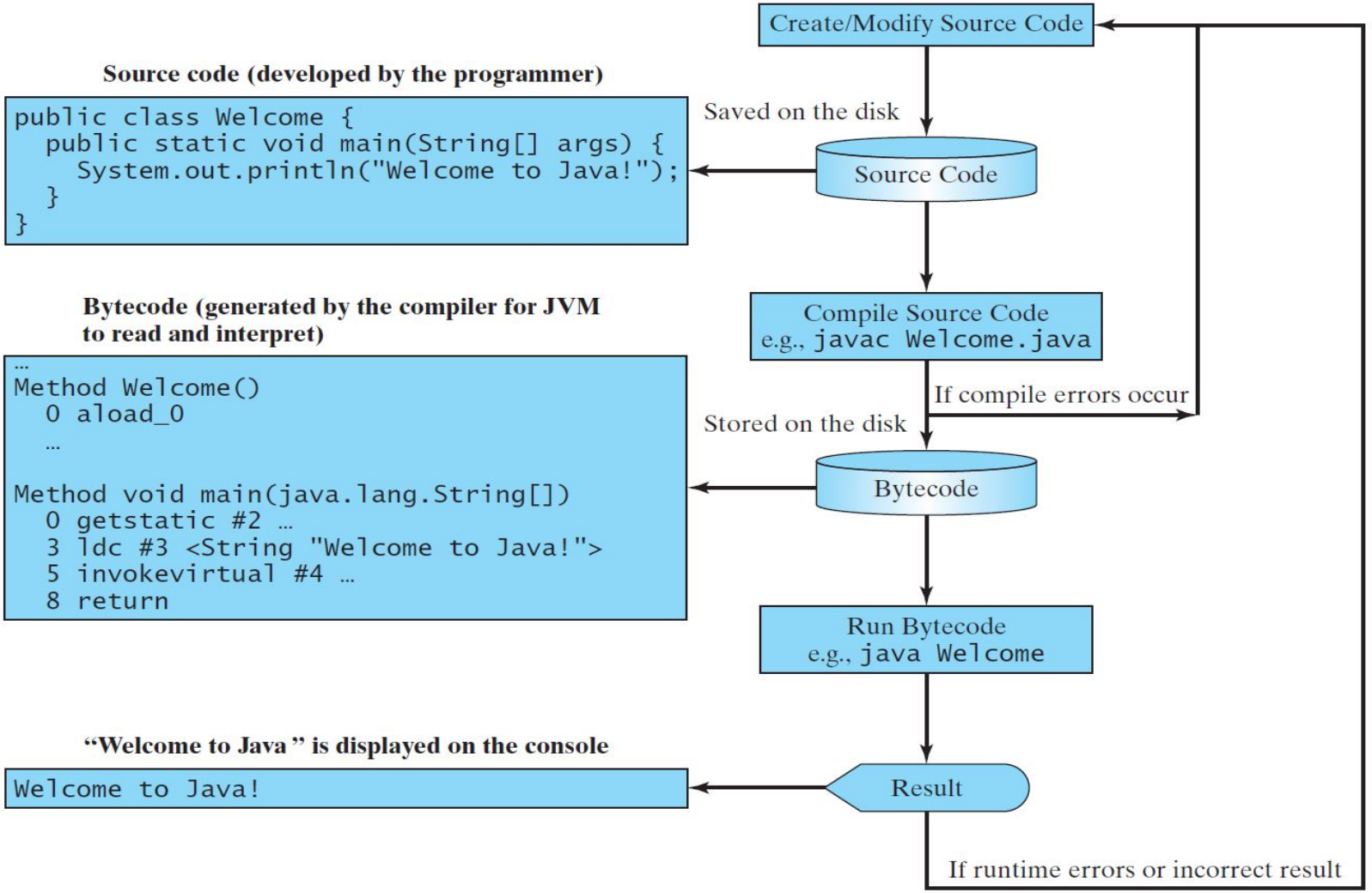
notepad Welcome.java

from the DOS prompt.



Creating, Compiling, and Running Programs

You save a Java program in a **.java file** and compile it into a **.class file**. The **.class file** is executed by the **Java Virtual Machine**.

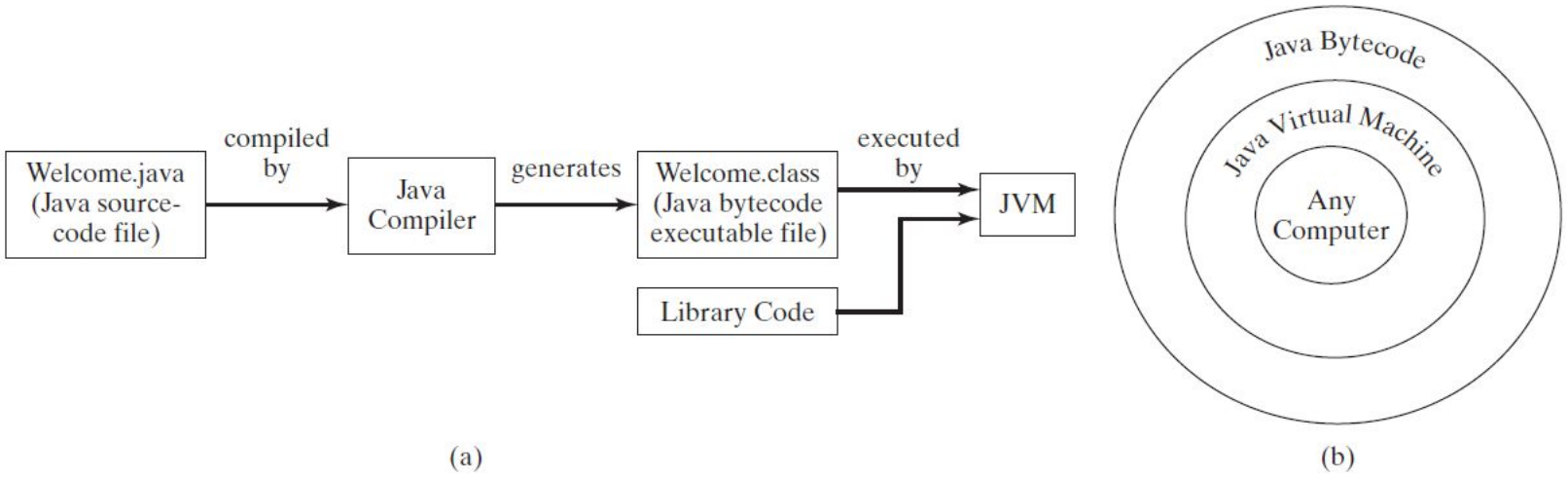


Compiling Java Source Code

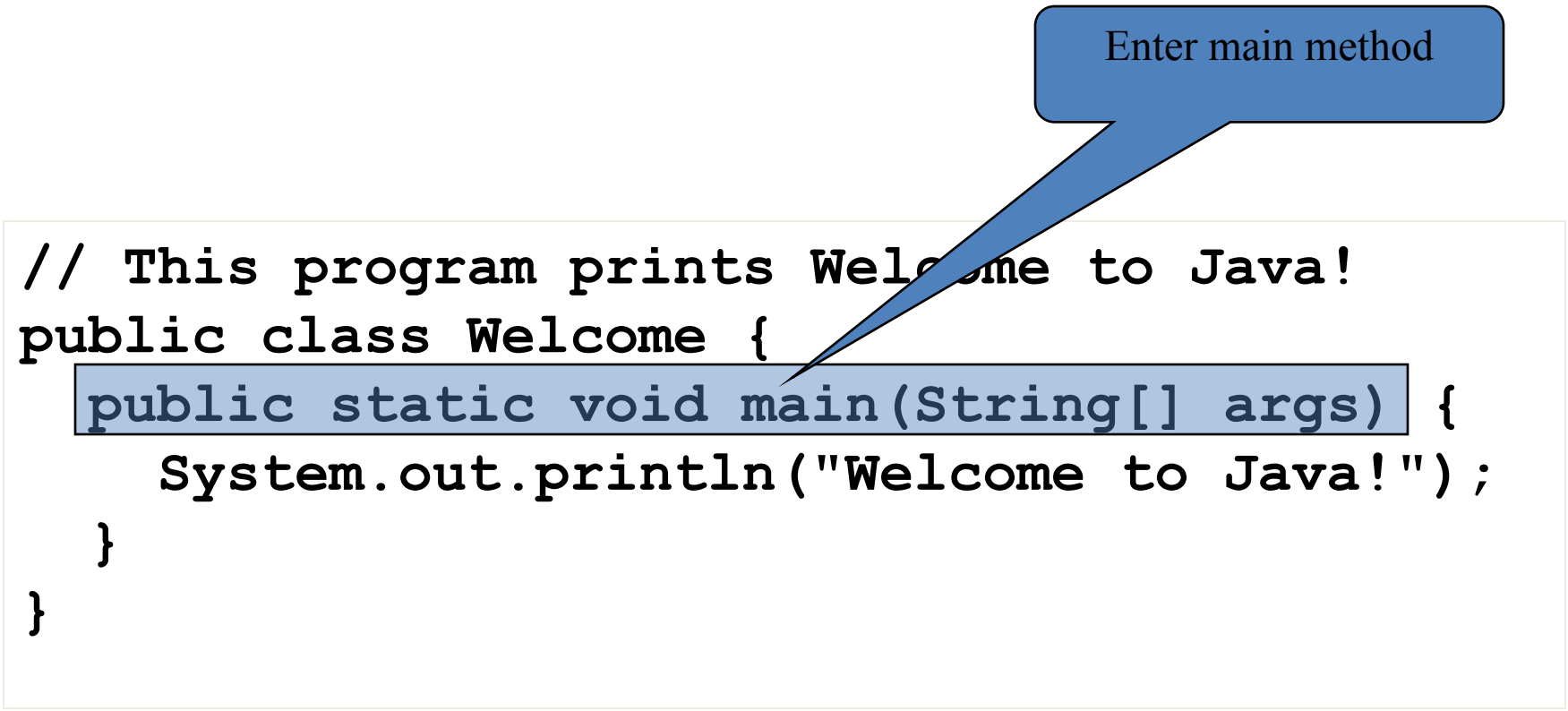
With Java, you write the program once, and compile the source program into a special type of object code, known as *bytecode*.

The bytecode can then run on any computer with a Java Virtual Machine, as shown below.

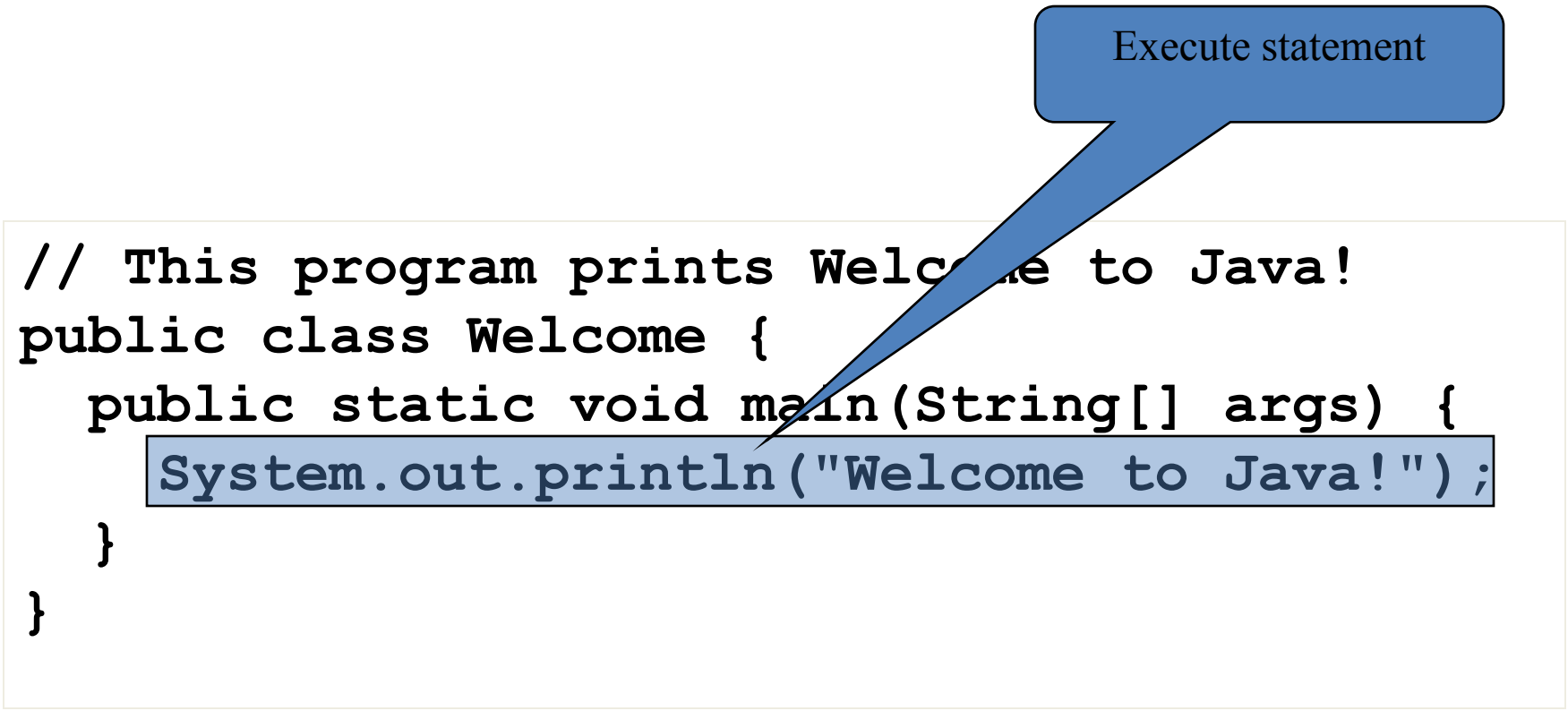
Java Virtual Machine is a software that interprets Java bytecode.



Trace a Program Execution



Trace a Program Execution

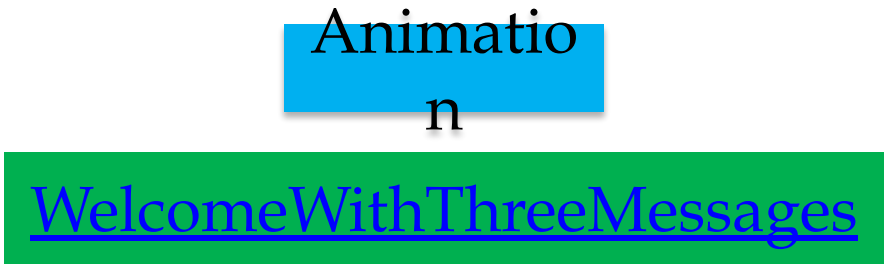


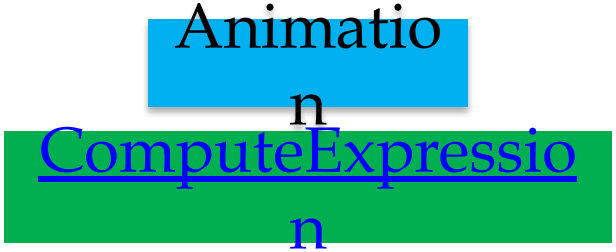
Trace a Program Execution

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

print a message to the console

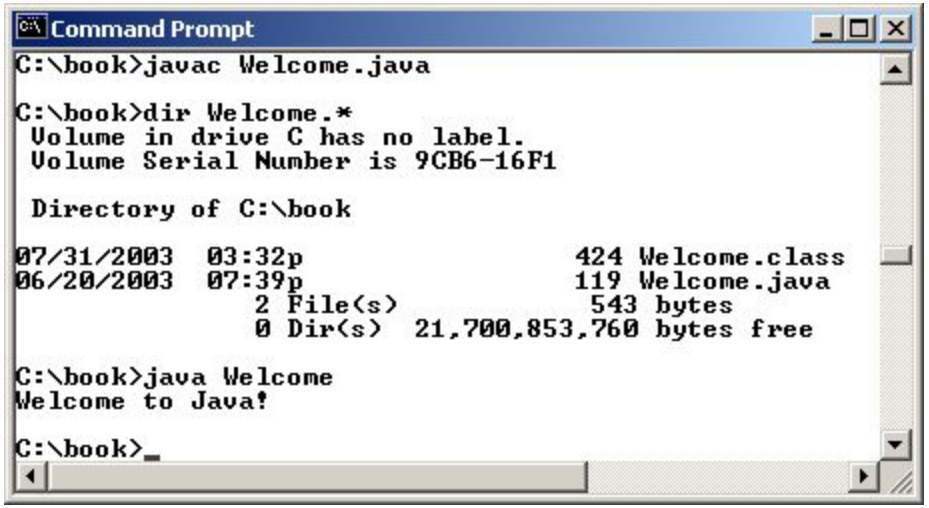
Two More Simple Examples





Compiling and Running Java from the Command Window

- Set path to JDK bin directory
  - set path=c:\Program Files\java\jdk1.8.0\bin
- Set classpath to include the current directory
  - set classpath=.
- Compile
  - javac Welcome.java
- Run
  - java Welcome



Anatomy of a Java Program

- Class name
- Main method
- Statements
- Statement terminator
- Reserved words
- Comments
- Blocks

Class Name

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

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Main Method

In order to run a class, the class must contain a method named main. The program is executed from the main method.

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Statement

A statement represents an action or a sequence of actions. The statement System.out.println("Welcome to Java!") in the program is a statement to display the message "Welcome to Java!“.

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Statement Terminator

Every statement in Java ends with a semicolon (;).

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

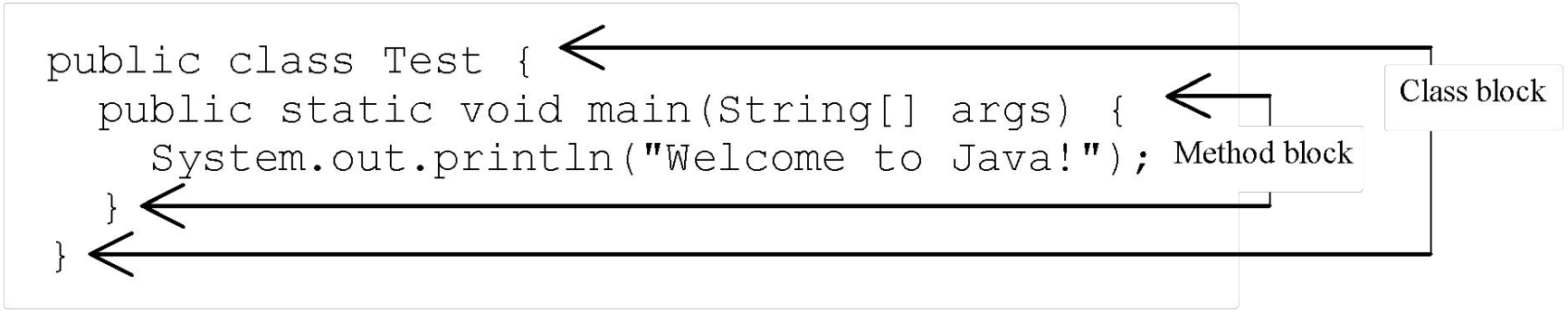
Reserved words

Reserved words or keywords are words that 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.

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Blocks

A pair of braces in a program forms a block that groups components of a program.



Special Symbols

Character	Name	Description
{}	Opening and closing braces	Denotes a block to enclose statements.
()	Opening and closing parentheses	Used with methods.
[]	Opening and closing brackets	Denotes an array.
//	Double slashes	Precedes a comment line.
" "	Opening and closing quotation marks	Enclosing a string (i.e., sequence of characters).
;	Semicolon	Marks the end of a statement.

{ ... }

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

(...)

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

;

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

// ...

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

" "
...

```
// This program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Programming Style

Good programming style make a program easy to read and help programmers prevent errors.

- Appropriate Comments
- Naming Conventions
- Proper Indentation and Spacing Lines
- Block Styles

Appropriate Comments

Include a **summary at the beginning of the program** to explain what the program does, its key features, its supporting data structures, and any unique techniques it uses.

Include your name, class section, instructor, date, and a brief description at the beginning of the program.

Naming Conventions

- Choose meaningful and descriptive names.
- Class names:
  - Capitalize the first letter of each word in the name. For example, the class name `ComputeExpression`.

Proper Indentation and Spacing

- Indentation
  - Indent two spaces.

- Spacing
  - Use blank line to separate segments of the code.

Block Styles

Use end-of-line style for braces.

Next-line style

```
public class Test
{
  public static void main(String[] args)
  {
    System.out.println("Block Styles");
  }
}
```

End-of-line style

```
public class Test {
  public static void main(String[] args) {
    System.out.println("Block Styles");
  }
}
```

Programming Errors

Programming errors can be categorized into three types:

- Syntax Errors
  - Detected by the compiler
- Runtime Errors
  - Cause the program to abort
- Logic Errors
  - Produce incorrect result

Syntax Errors

```
public class ShowSyntaxErrors {
  public static main(String[] args) {
    System.out.println("Welcome to Java);
  }
}
```

```
Error is: " is absent.
```

ShowSyntaxErrors

Runtime Errors

```
public class ShowRuntimeErrors {
  public static void main(String[] args) {
    System.out.println(1 / 0);
  }
}
```

```
Error is: Division by zero.
```

ShowRuntimeErrors

Logic Errors

```
public class ShowLogicErrors {
  public static void main(String[] args) {
    System.out.println("Celsius 35 is Fahrenheit degree ");
    System.out.println((9 / 5) * 35 + 32);
  }
}
```

Error is: 9 / 5 division is in integer; it must be 9.0 / 5 in decimal.

ShowLogicErrors

Compiling and Running Java from Eclipse