



Chapter 1

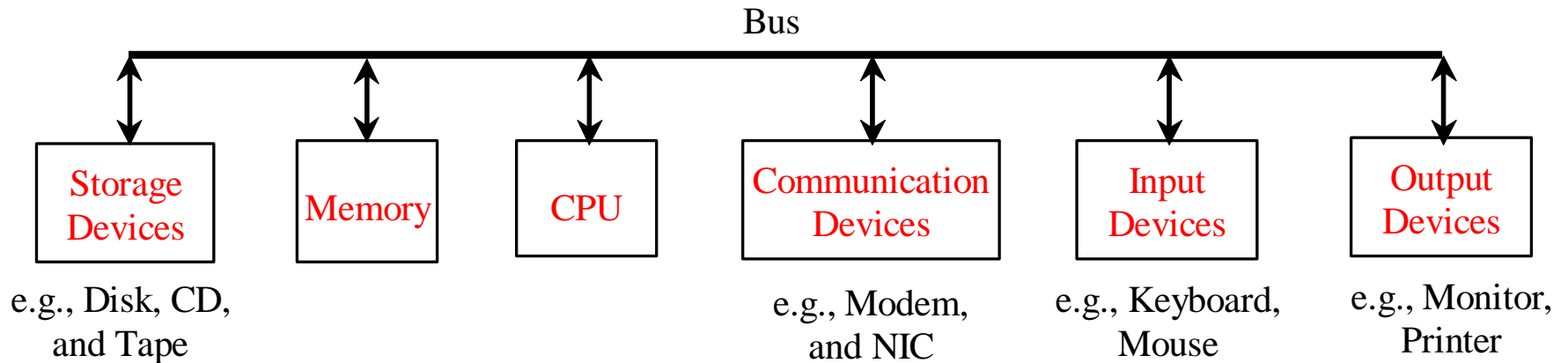
Introduction to Computers, Programs, and Java

Contents

- ➡ What is a Computer
- ➡ Computer Organization
- ➡ Different types of programming language
- ➡ Computer system (Input-Process-Output)
- ➡ The java language specification, API, JDK and IDE
- ➡ A simple java program
- ➡ Creating, compiling and executing a Java program
- ➡ Programming style and documentation
- ➡ Programming Errors and debugging

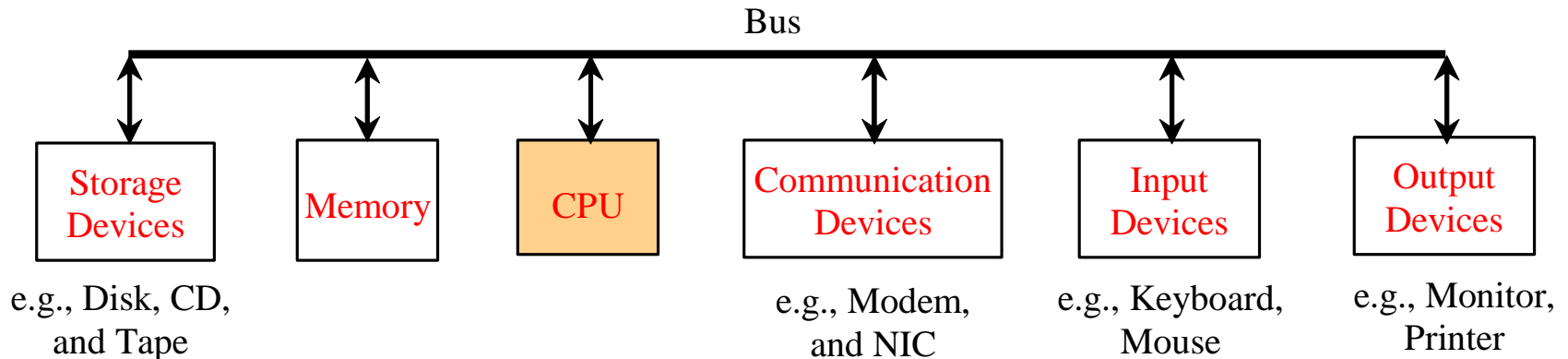
What is a Computer?

A computer consists of a CPU, memory, hard disk, floppy disk, monitor, printer, and communication devices.



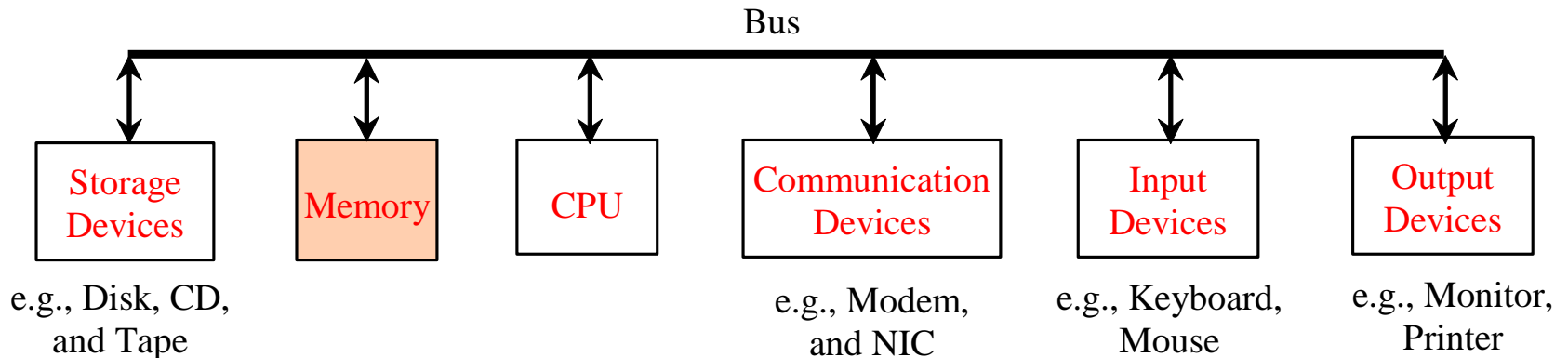
CPU

The central processing unit (CPU) is the brain of a computer. It retrieves instructions from memory and executes them. The CPU speed is measured in megahertz (MHz), with 1 megahertz equaling 1 million pulses per second. The speed of the CPU has been improved continuously.



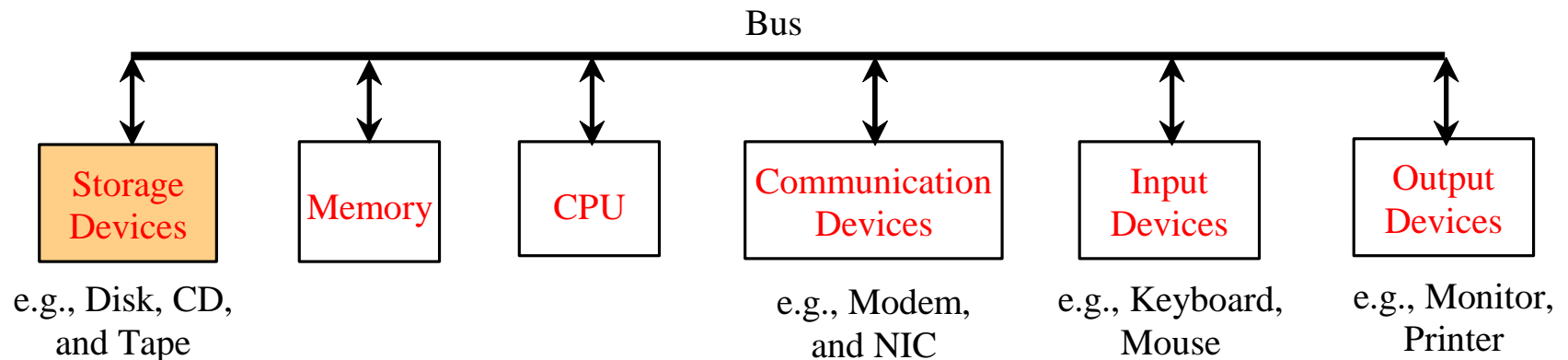
Memory

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. 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.



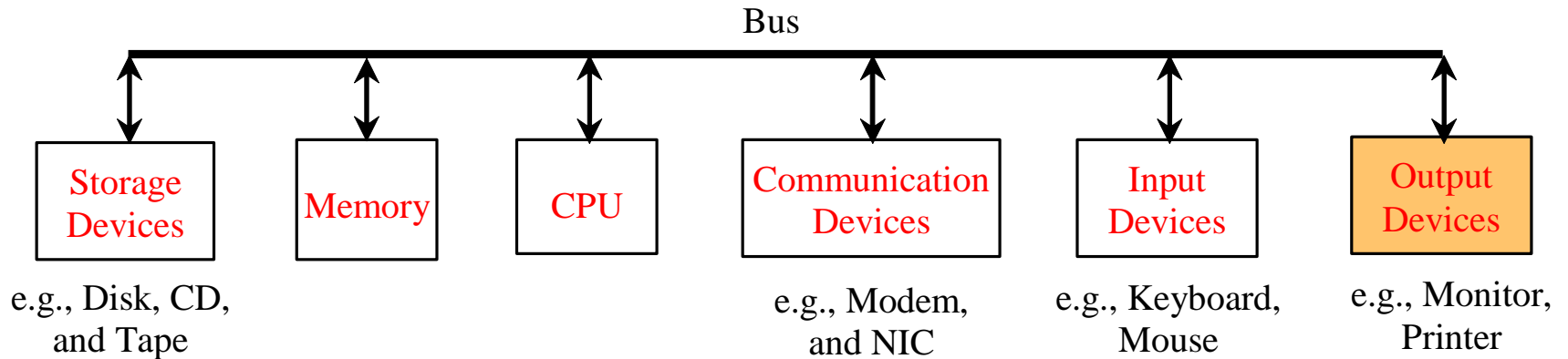
Storage Devices

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 USB (Universal Serial Bus) flash drive.



Output Devices: Monitor

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



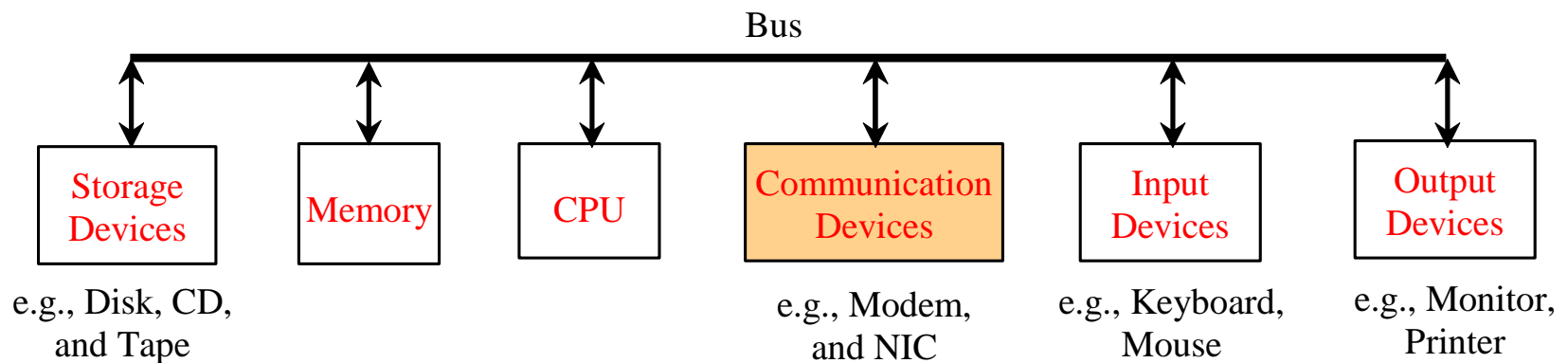
Monitor Resolution and Dot Pitch

resolution 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. A common resolution for a 17-inch screen, for example, is 1,024 pixels wide and 768 pixels high. The resolution can be set manually. The higher the resolution, the sharper and clearer the image is.

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

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 *10BaseT*, can transfer data at 10 mbps (million bits per second).



Programs

Computer *programs*, known as *software*, are instructions 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 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 native machine language is a tedious 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:

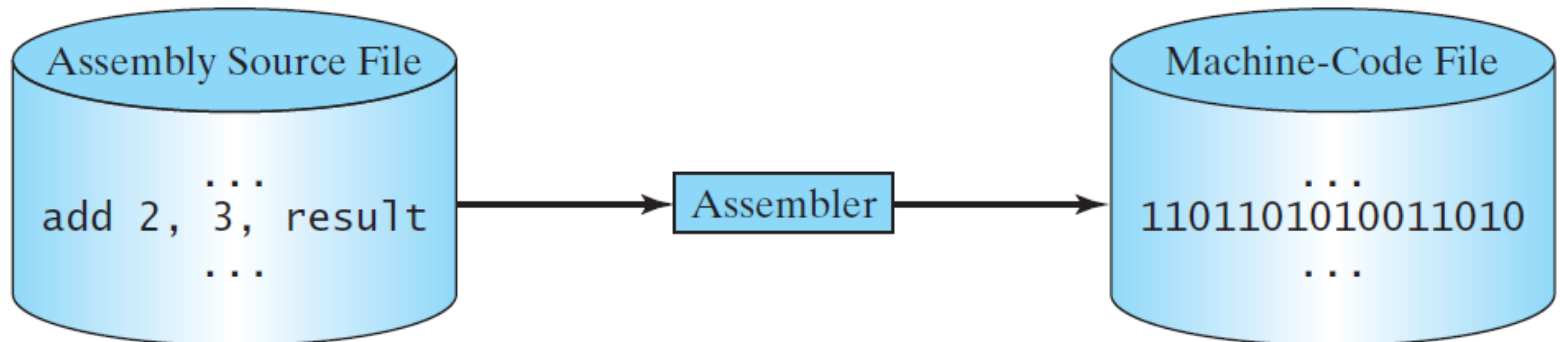
```
1101101010011010
```

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:

ADD 2, 3, result



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

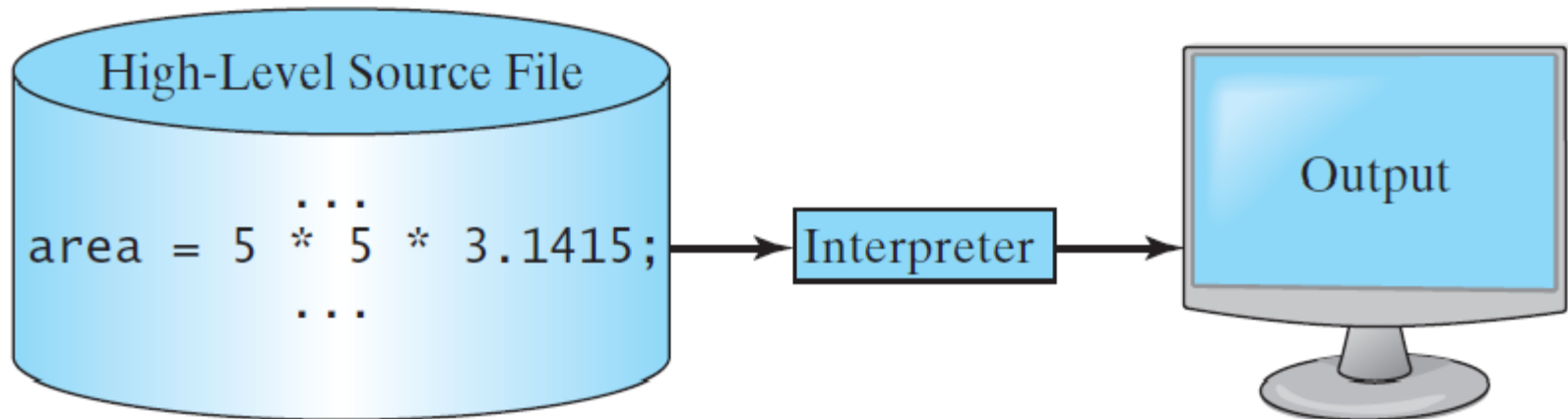
Language	Description
Ada	Named for Ada Lovelace, who worked on mechanical general-purpose computers. The Ada language was developed for the Department of Defense and is used mainly in defense projects.
BASIC	Beginner's All-purpose Symbolic Instruction Code. It was designed to be learned and used easily 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, who pioneered calculating machines in the seventeenth century. 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*.

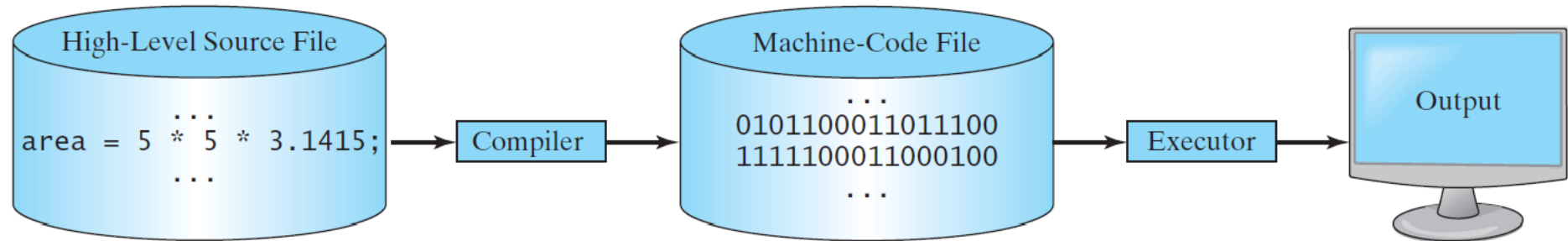
Interpreting Source Code

- An interpreter reads one statement from the source code
- Translates it to the machine code or virtual machine code, and then
- Executes it right away, as shown in the following figure.



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.



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. The future of computing is being profoundly 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.

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) a. k. a. JDK 5 or Java 5
- ☞ JDK 1.6 (2006) a. k. a. JDK 6 or Java 6
- ☞ JDK 1.7 (2011) a. k. a. JDK 7 or Java 7
- ☞ **JDK 1.8 (2014) a. k. a. JDK 8 or Java 8**
- ☞ JDK 16 (2021 March, April)
- ☞ JDK 17.0.1 (2021 September)

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

- ☞ JCreator
- ☞ NetBeans
- ☞ Eclipse

A Simple Java Program

Listing 1.1

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

Welcome

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

Source code (developed by the programmer)

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

Bytecode (generated by the compiler for JVM to read and interpret)

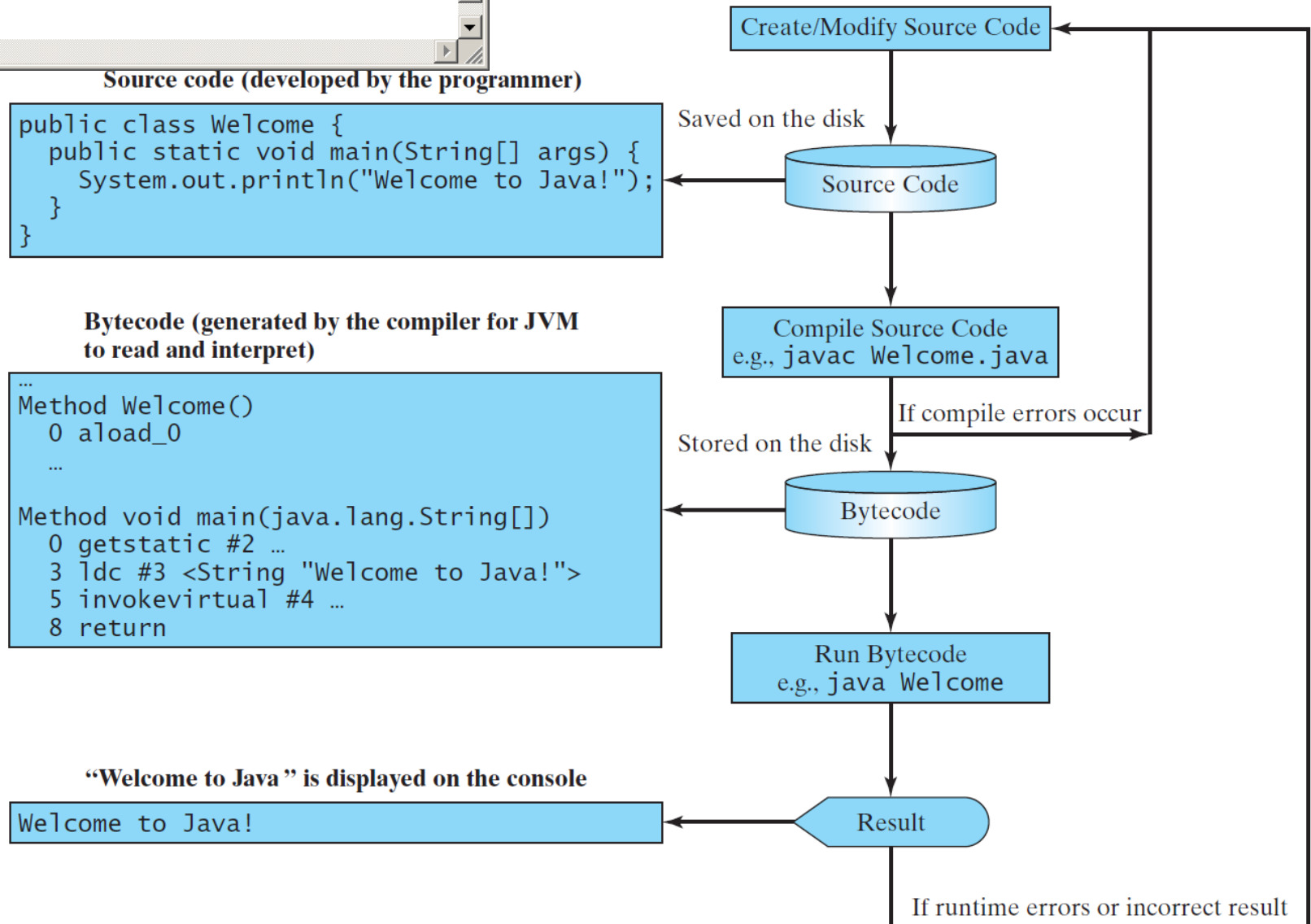
```
...
Method Welcome()
  0 aload_0
  ...

Method void main(java.lang.String[])
  0 getstatic #2 ...
  3 ldc #3 <String "Welcome to Java!">
  5 invokevirtual #4 ...
  8 return
```

“Welcome to Java” is displayed on the console

Welcome to Java!

Creating, Compiling, and Running Programs



Trace a Program Execution

Enter main method

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

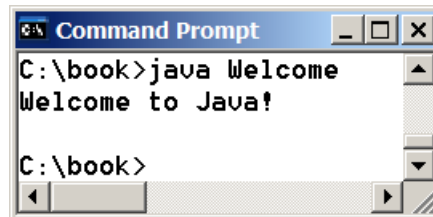
Trace a Program Execution

Execute statement

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

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

WelcomeWithThreeMessages

ComputeExpression

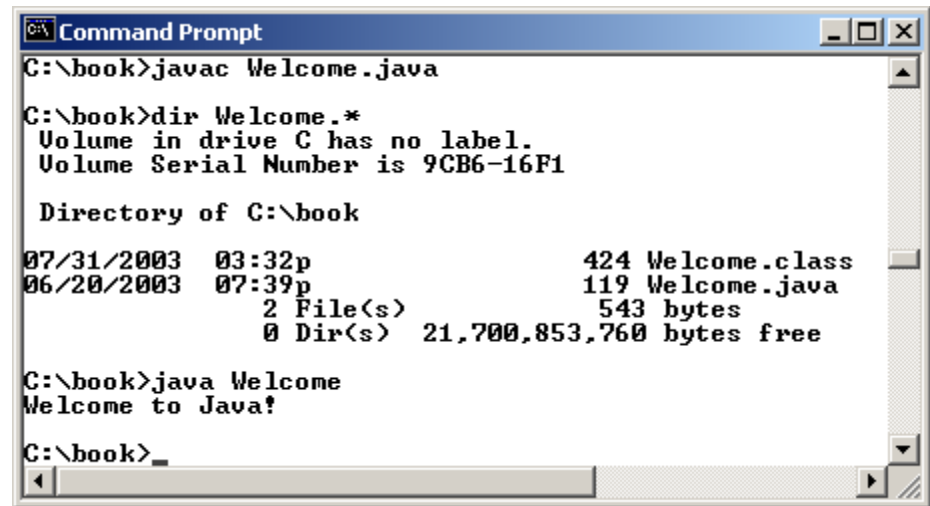
Supplements

☞ JCreator installer download link

☞ JDK download link

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



```
Command Prompt
C:\book>javac Welcome.java

C:\book>dir Welcome.*
Volume in drive C has no label.
Volume Serial Number is 9CB6-16F1

Directory of C:\book

07/31/2003  03:32p                424 Welcome.class
06/20/2003  07:39p                119 Welcome.java
                2 File(s)            543 bytes
                0 Dir(s)  21,700,853,760 bytes free

C:\book>java Welcome
Welcome to Java!

C:\book>
```

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, 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

- ☞ Line 2 defines the 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 greeting "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.

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

Class block

Method block

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.

{ ... }

Braces

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

(...)

Parentheses

```
// 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 and Documentation

- ☞ 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 student identity, name, batch number, submission date, and a brief description (question) 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.

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

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

☞ Syntax Errors

- Detected by the compiler

☞ Runtime Errors

- Causes the program to abort

☞ Logic Errors

- Produces incorrect result

Syntax Errors

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

ShowSyntaxErrors

Runtime Errors

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

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);  
    }  
}
```

ShowLogicErrors

Compiling and Running Java from JCreator