



TM105: Introduction to Computer Programming

Meeting 1 (Chapter 1)

Introduction to Computers, Programs, and Java

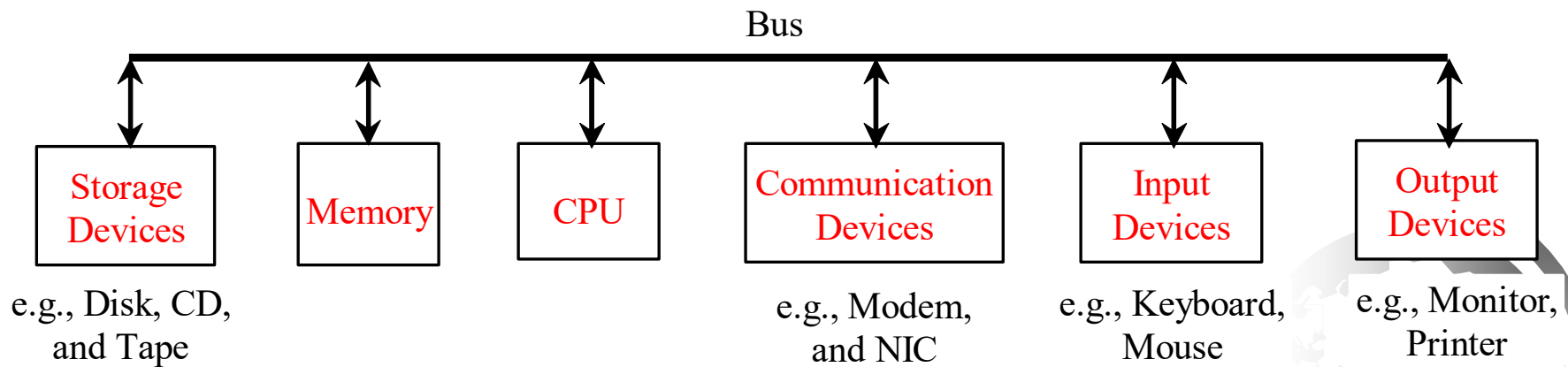
Objectives

- ☞ To understand computer basics, programs, and operating systems.
- ☞ To describe the relationship between Java and the World Wide Web.
- ☞ To write a simple Java program.
- ☞ To explain the basic syntax of a Java program.
- ☞ To create, compile, and run Java programs.
- ☞ To develop Java programs using NetBeans.



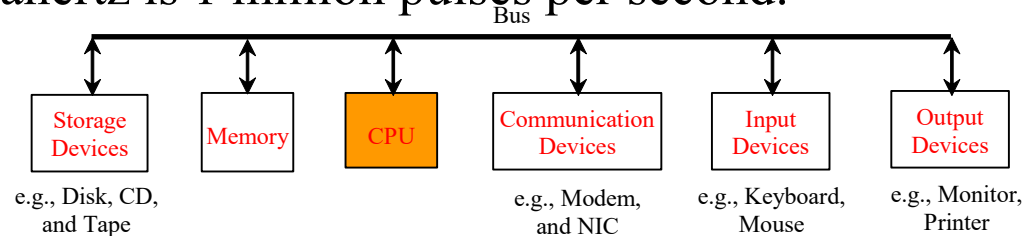
What is a Computer?

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).
 - 1 megahertz is 1 million pulses per second.



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)
 - Tape drives.



Output Devices: Monitor

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



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

1101101010011010

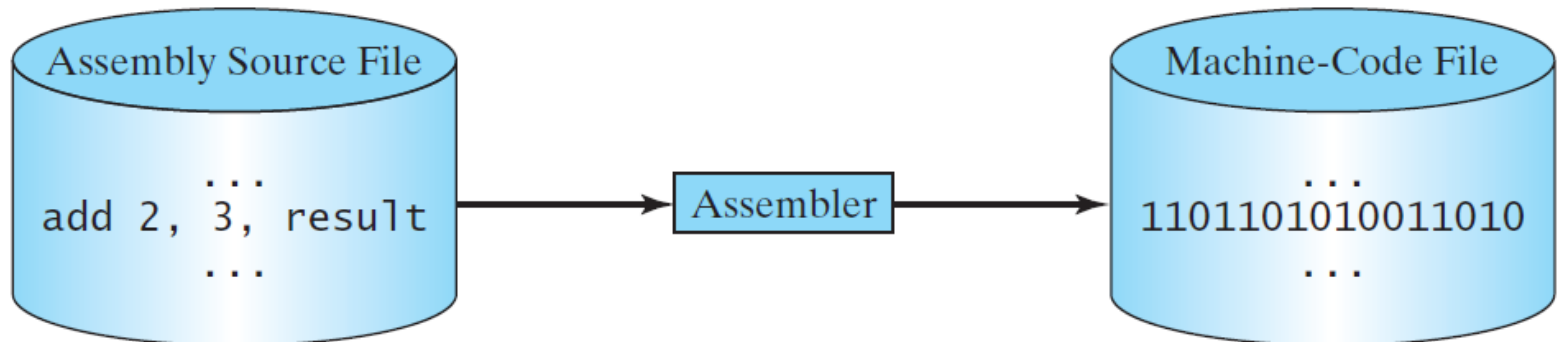


Programming Languages

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, 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.

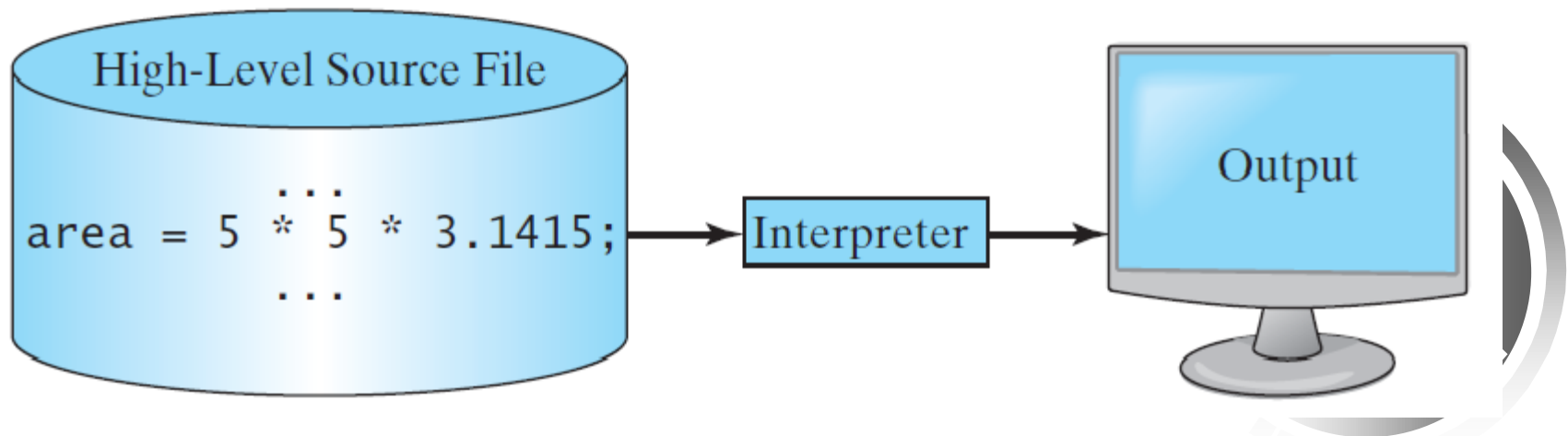
Interpreting/Compiling Source Code

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



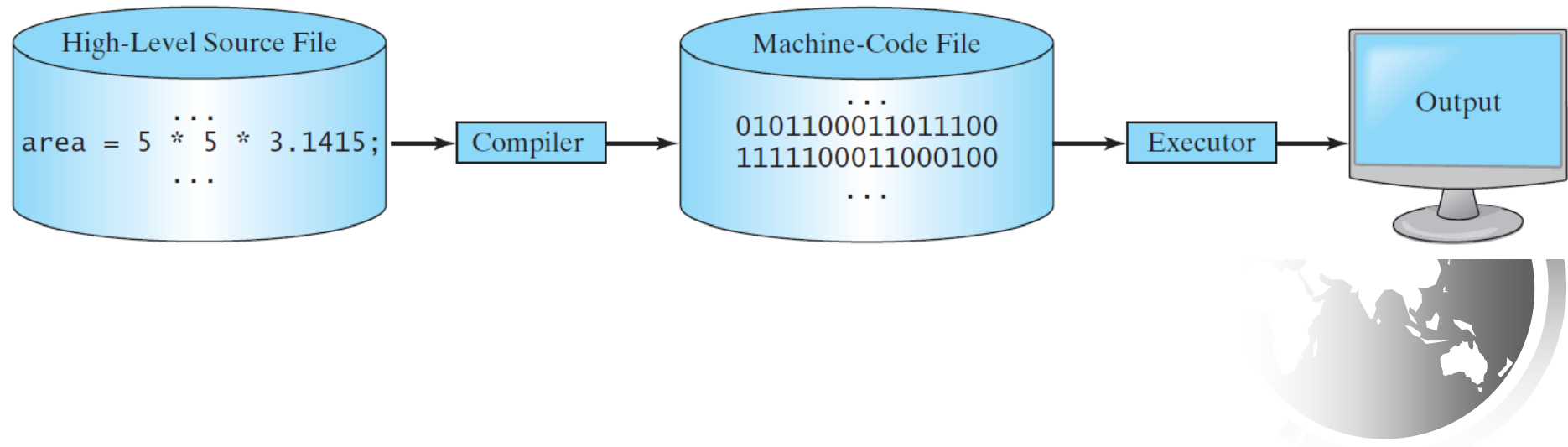
Interpreting Source Code

- 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.
- A statement from the source code may be



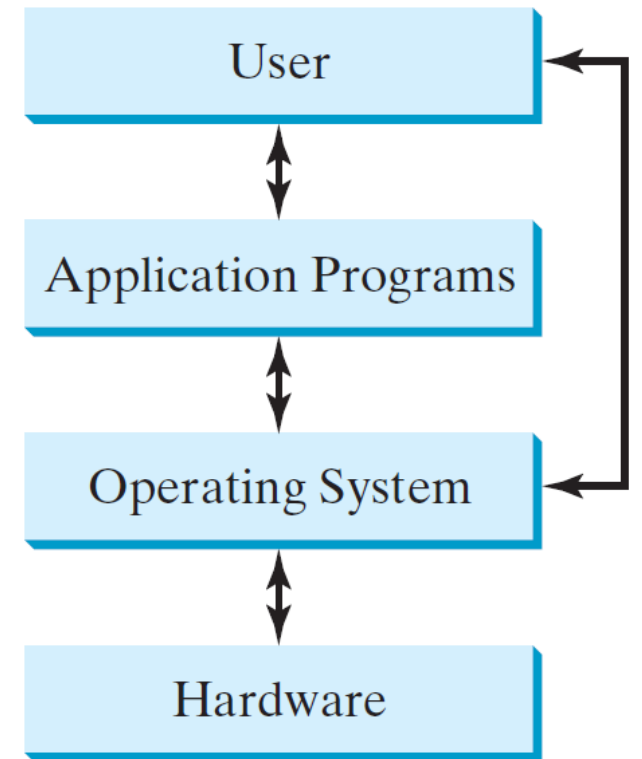
Compiling Source Code

- A compiler translates the entire source code into a machine-code file.
- The machine-code file is then executed.



Operating Systems

- The operating system (OS) is a program that manages and controls a computer's activities.
- Popular desktop operating systems include Linux, Windows 7 and Mac OS X.
- Popular mobile operating systems used in smartphones and tablets include Google's Android, BlackBerry OS and Apple's iOS (for iPhone and iPad).
- Application programs, such as a Web browser or a word processor, cannot run unless an operating system is installed and running on the computer.



Why Java?

- ☞ Java is the world's most widely used computer programming language.
- ☞ In use today are more than a billion general-purpose computers and billions more Java-enabled cell phones, smartphones and handheld devices (such as tablet computers).
- ☞ Java is a general purpose programming language.
- ☞ Java is the Internet programming language.
- ☞ Java is **Object-Oriented** (OO) — today's key programming methodology.



Why Java?

- 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.



Characteristics of Java

- Java Is Simple
- Java Is Object-Oriented
- Java Is Distributed
- Java Is Interpreted
- Java Is Robust
- Java Is Secure
- Java Is Architecture-Neutral
- Java Is Portable
- Java's Performance
- Java Is Multithreaded
- Java Is Dynamic

www.cs.armstrong.edu/liang/JavaCharacteristics.pdf



Java Programs

- ▶ Java programs may be divided into:
 - Standalone/desktop applications (do not need a web browser to run)
 - Applets (need a web browser to run)
- ▶ To write Java programs, you need to download and install the Java Development Kit (JDK), which is a development environment includes tools useful for this purpose.
- ▶ JDK could be found in various editions:
 - Java Standard Edition (Java SE)
 - Java Enterprise Edition (Java EE)
 - Java Micro Edition (Java ME)



Java Editions

☞ Java Standard Edition (Java SE)

- Used for developing **cross-platform**, general-purpose applications.
- You can download the JDK and its documentation from www.oracle.com/technetwork/java/javase/downloads/index.html

☞ Java Enterprise Edition (Java EE)

- Geared toward developing large-scale, distributed networking applications and web-based applications.

☞ Java Micro Edition (Java ME)

- Geared toward developing applications for small, memory-constrained devices, such as BlackBerry smartphones.

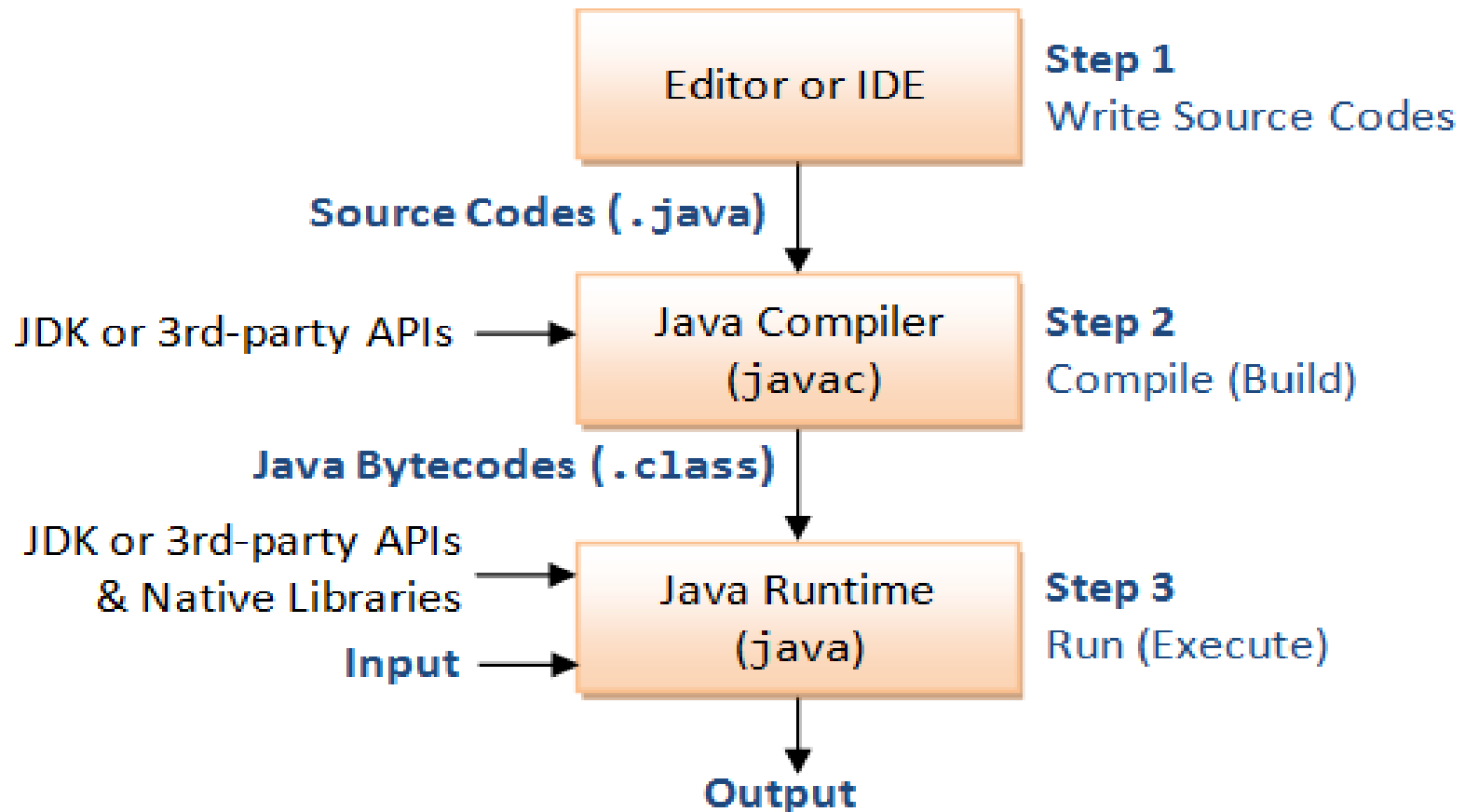


Java and a Typical Java Development Environment

- Java programs normally go through five phases
 - **edit**
 - using a text editor like Notepad or an IDE like NetBeans to write the program
 - saving the program (source code) in a **.java** file
 - **compile**
 - Using the command **javac** to create a **.class** file, which contains the compiled version of the program (Java byte code)
 - load
 - verify
 - **execute**
 - Using the command **java**, where Java Virtual Machine (**JVM**) executes the Java byte code in the **.class** file
- You can visit Oracle's New to Java Center at:

www.oracle.com/technetwork/topics/newtojava/overview/index.html

Java and a Typical Java Development Environment



Popular Java IDEs

☞ NetBeans

☞ Eclipse



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!");  
    }  
}
```

Welcome

Run

Note: Clicking the green button displays the source code with interactive animation. You can also run the code in a browser. Internet connection is needed for this button.

Note: Clicking the blue button runs the code from Windows. If you cannot run the buttons, see liveexample.pearsoncmg.com/slide/javaslidenote.doc.

Trace a Program Execution


Enter main method

```
// This program prints Welcome to Java!  
public class Welcome {  
    {  
        System.out.println("Welcome to Java!");  
    }  
}
```




Trace a Program Execution

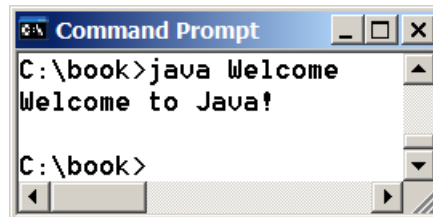
Execute statement

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
          
    }  
}
```



Trace a Program Execution

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
          
    }  
}
```



```
Command Prompt  
C:\book>java Welcome  
Welcome to Java!  
C:\book>
```

print a message to the
console

Two More Simple Examples

WelcomeWithThreeMessages

Run

ComputeExpression

Run



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

```
// This program prints Welcome to Java!  
is welcome.  
public class  {  
    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!")` 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.
- Keywords are always spelled with all lowercase letters.

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



Reserved words

Java Keywords				
<code>abstract</code>	<code>assert</code>	<code>boolean</code>	<code>break</code>	<code>byte</code>
<code>case</code>	<code>catch</code>	<code>char</code>	<code>class</code>	<code>continue</code>
<code>default</code>	<code>do</code>	<code>double</code>	<code>else</code>	<code>enum</code>
<code>extends</code>	<code>final</code>	<code>finally</code>	<code>float</code>	<code>for</code>
<code>if</code>	<code>implements</code>	<code>import</code>	<code>instanceof</code>	<code>int</code>
<code>interface</code>	<code>long</code>	<code>native</code>	<code>new</code>	<code>package</code>
<code>private</code>	<code>protected</code>	<code>public</code>	<code>return</code>	<code>short</code>
<code>static</code>	<code>strictfp</code>	<code>super</code>	<code>switch</code>	<code>synchronized</code>
<code>this</code>	<code>throw</code>	<code>throws</code>	<code>transient</code>	<code>try</code>
<code>void</code>	<code>volatile</code>	<code>while</code>		
<i>Keywords that are not currently used</i>				
<code>const</code>	<code>goto</code>			

Fig. C.1 | Java keywords.

Java also contains the reserved words `true` and `false`, which are boolean literals, and `null`, which is the literal that represents a reference to nothing. Like keywords, these reserved words cannot be used as identifiers.

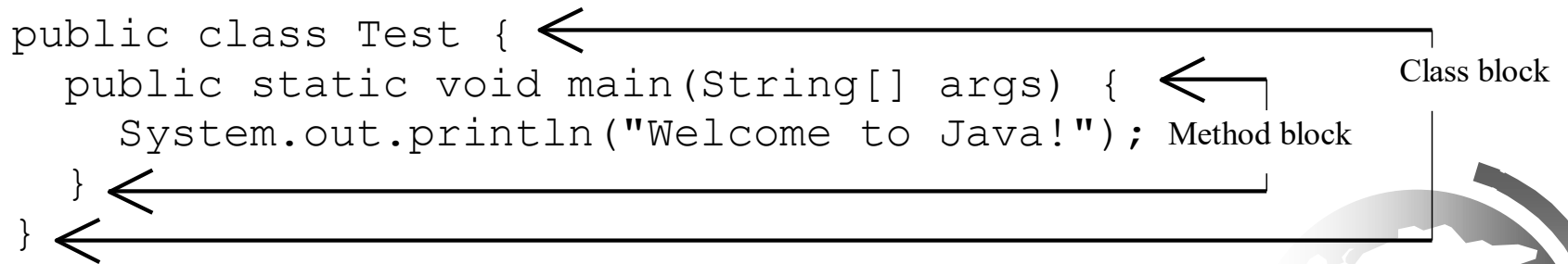
Blocks

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.



{ ... }

```
// 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 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 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");
    }
}
```



Thanks!

