Chapter 1 Introduction to Programs, and Java

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:
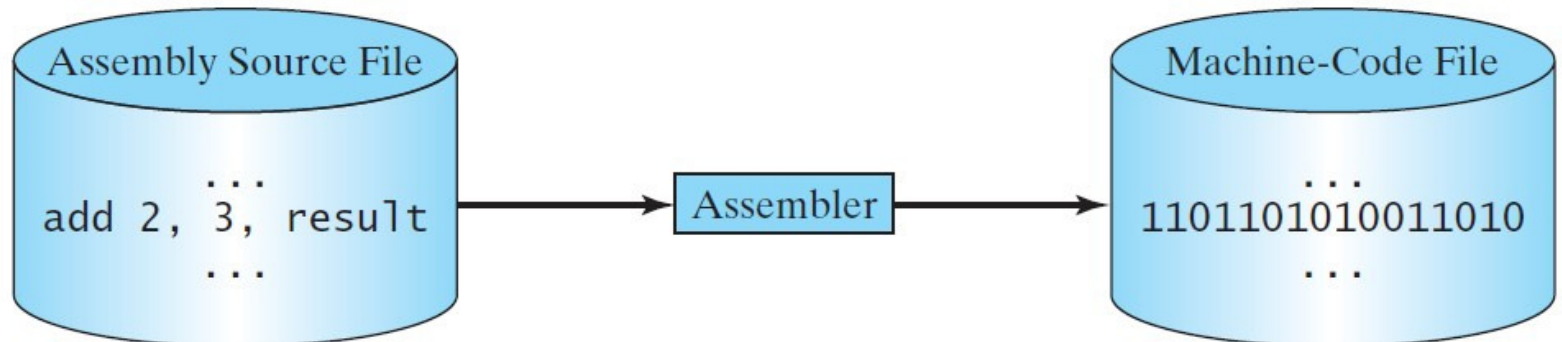
1101101010011010

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:

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 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.
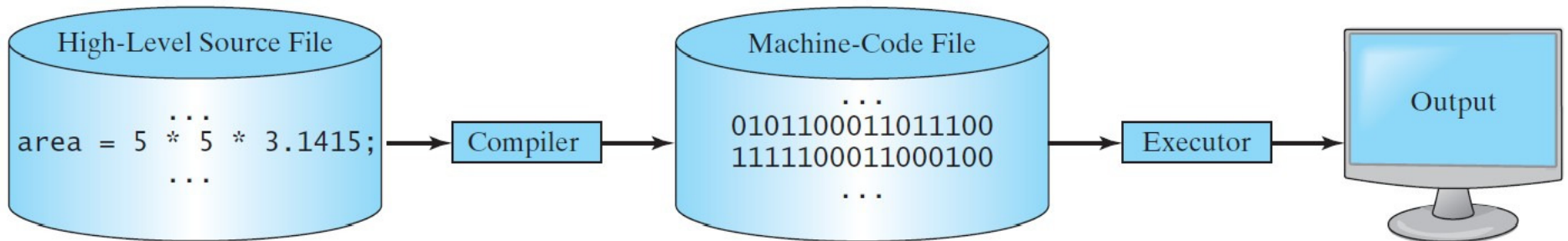
Interpreting Source Code

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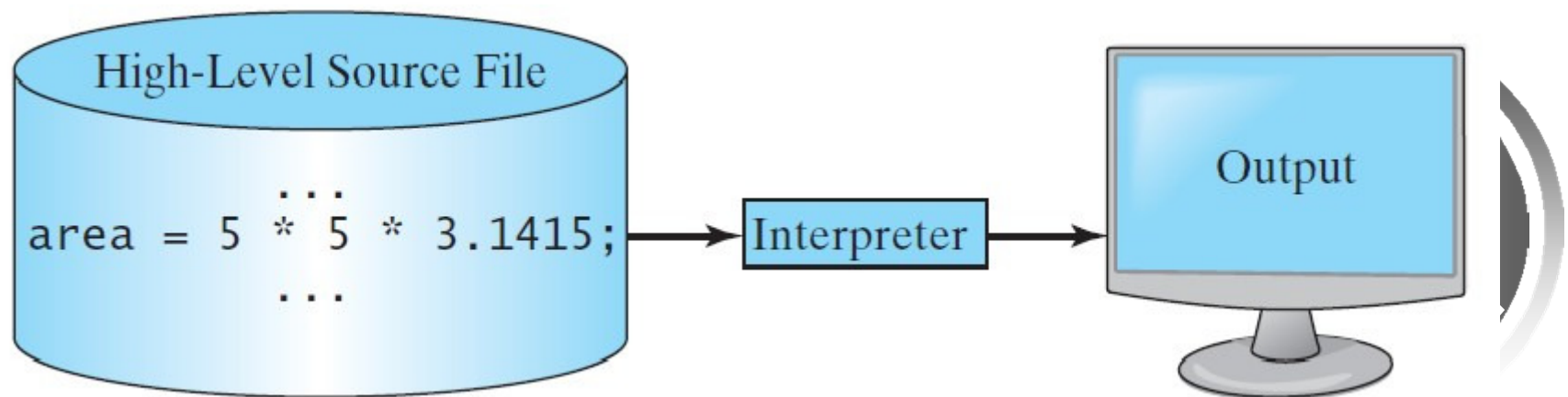
Compiling Source Code

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.

High-Level Source File		Machine-Code File		
. . . area = 5 * 5 * 3.1415; . . .	→ Compiler →	. . . 0101100011011100 1111100011000100 . . .	→ Executor →	Output

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.
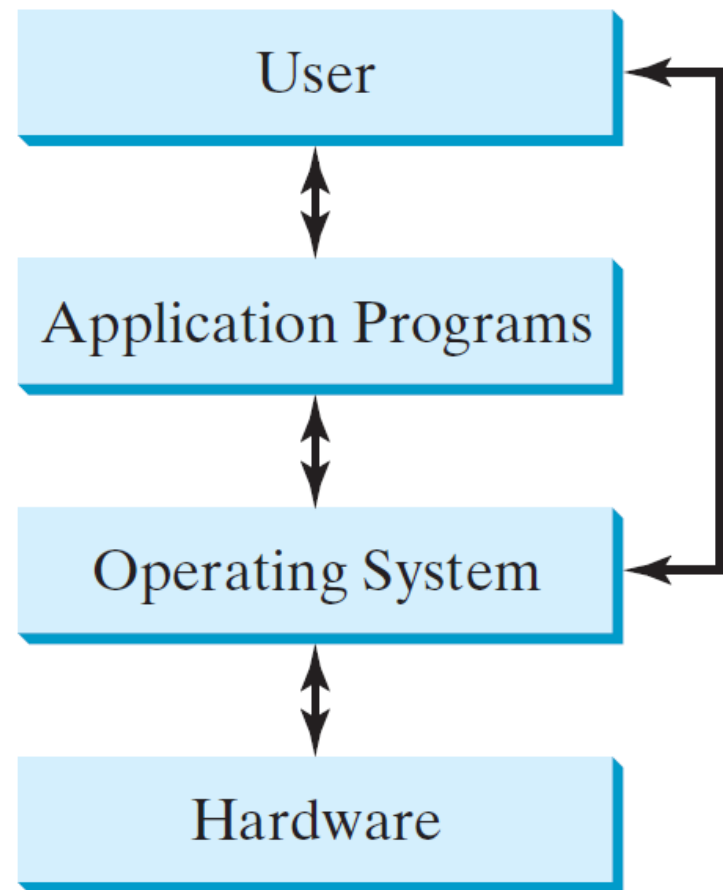
Why Java?

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.

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

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

Java is partially modeled on C++, but greatly simplified and improved. Some people refer to Java as "C++--" because it is like C++ but with more functionality and fewer negative aspects.
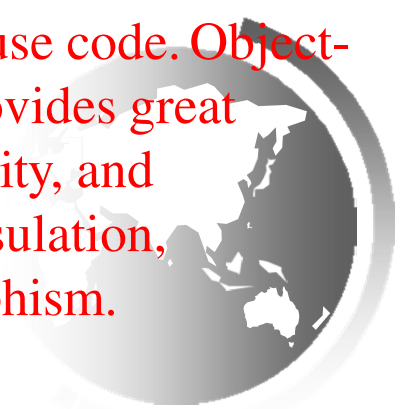
Characteristics of Java

- 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

Java is inherently object-oriented. Although many object-oriented languages began strictly as procedural languages, Java was designed from the start to be object-oriented. Object-oriented programming (OOP) is a popular programming approach that is replacing traditional procedural programming techniques.

One of the central issues in software development is how to reuse code. Object-oriented programming provides great flexibility, modularity, clarity, and reusability through encapsulation, inheritance, and polymorphism.

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

Distributed computing involves several computers working together on a network. Java is designed to make distributed computing easy. Since networking capability is inherently integrated into Java, writing network programs is like sending and receiving data to and from a file.

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

You need an interpreter to run Java programs. The programs are compiled into the Java Virtual Machine code called bytecode. The bytecode is machine-independent and can run on any machine that has a Java interpreter, which is part of the Java Virtual Machine (JVM).

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

Java compilers can detect many problems that would first show up at execution time in other languages.

Java has eliminated certain types of error-prone programming constructs found in other languages.

Java has a runtime exception-handling feature to provide programming support for robustness.

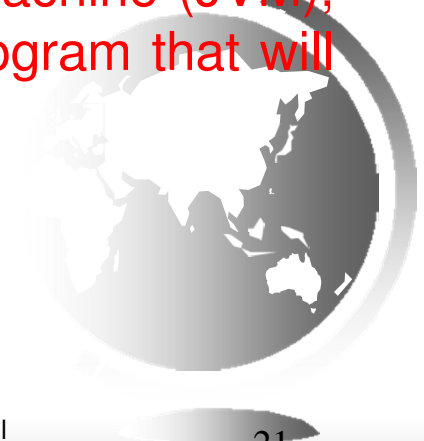# Characteristics of Java

- Java Is Simple

- Java Is Object-Oriented

- Java Is Distributed

- Java Is Interpreted

- Java Is Robust

- Java Is Secure   Java implements several security mechanisms to protect your system against harm caused by stray programs.

- Java Is Architecture-Neutral

- Java Is Portable

- Java's Performance

- Java Is Multithreaded

- Java Is Dynamic

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   Write once, run anywhere
- Java Is Portable
- Java's Performance   With a Java Virtual Machine (JVM), you can write one program that will run on any platform.
- Java Is Multithreaded
- Java Is Dynamic

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
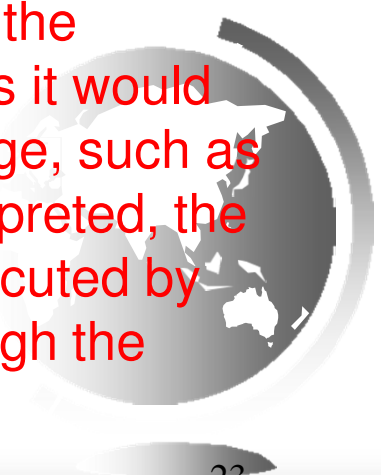- Java's Performance
- Java Is Multithreaded
- Java Is Dynamic

Because Java is architecture neutral, Java programs are portable. They can be run on any platform without being recompiled.

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

Java's performance is sometimes criticized. The execution of the bytecode is never as fast as it would be with a compiled language, such as C++. Because Java is interpreted, the bytecode is not directly executed by the system, but is run through the interpreter.

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

Multithread programming is smoothly integrated in Java, whereas in other languages you have to call procedures specific to the operating system to enable multithreading.

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

Java was designed to adapt to an evolving environment. New code can be loaded on the fly without recompilation. There is no need for developers to create, and for users to install, major new software versions. New features can be incorporated transparently as needed.

JDK Versions

java development kit

- 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 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.

☐

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

**Integrated development environment**

☐ NetBeans

☐ Eclipse

A Simple Java Program
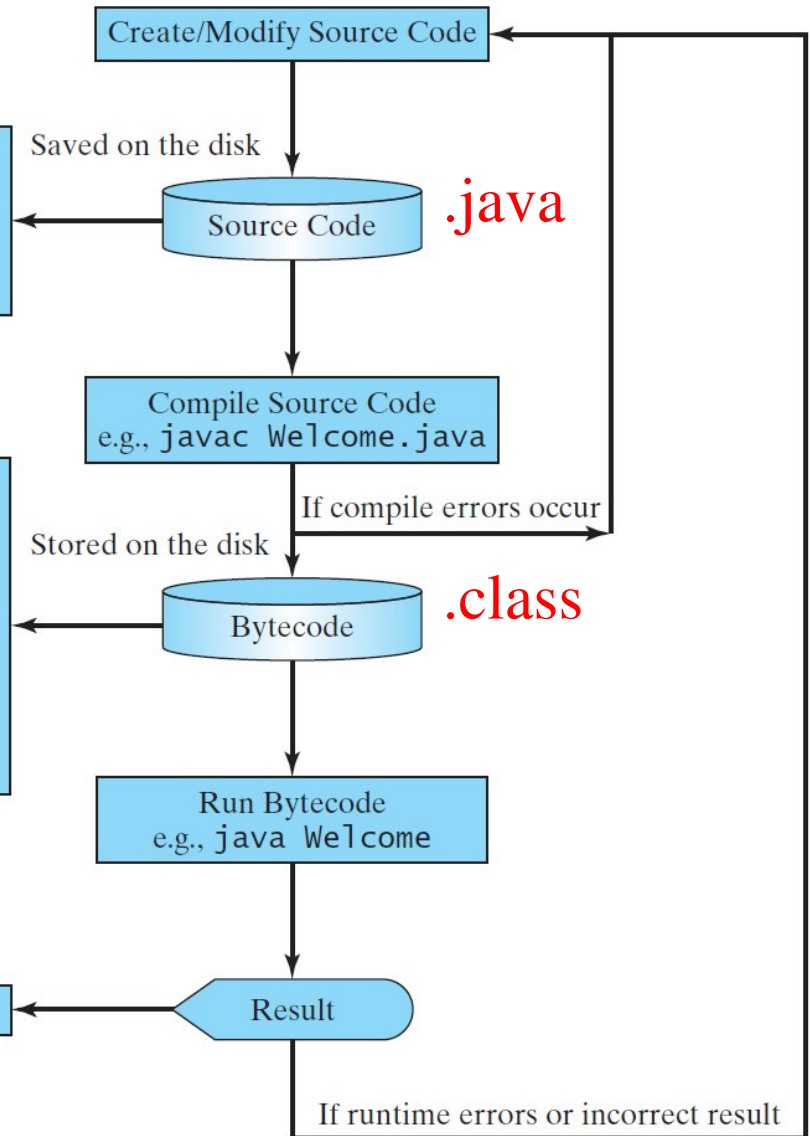
Listing 1.1

```java
// This program prints Welcome to Java!

public class Welcome {

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

  }

}
```

Creating and Editing Using NotePad

To use NotePad, type
notepad
Welcome.java
from the DOS prompt.

Command Prompt

```
C:\book>notepad Welcome.java_
```

Welcome - Notepad

File Edit Format View Help

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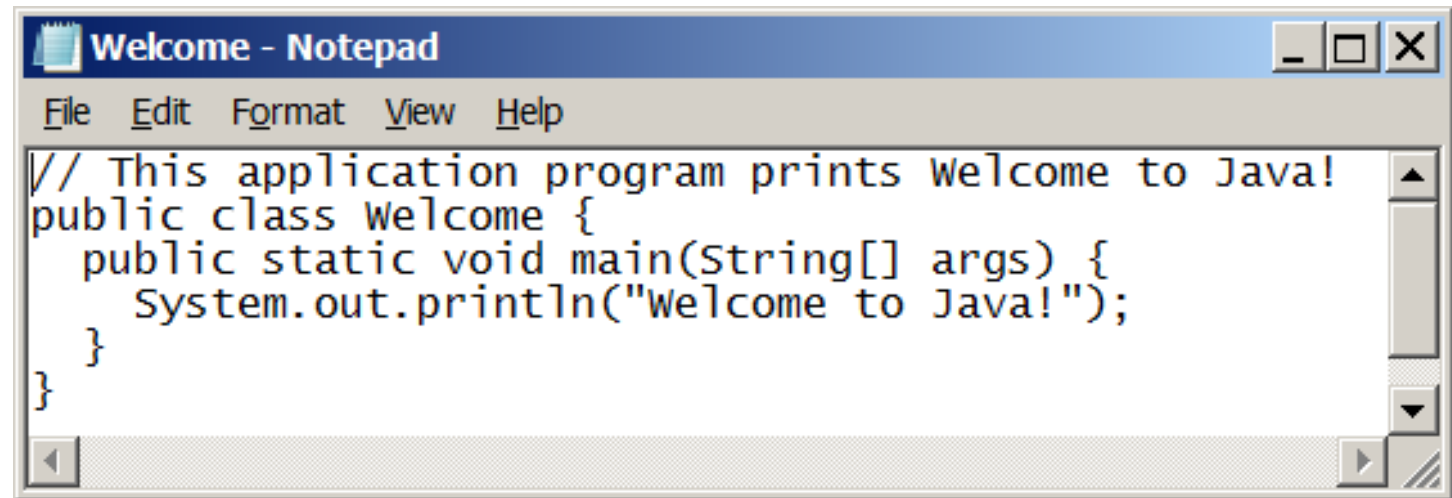
Creating, Compiling, and Running Programs

Welcome - Notepad

File Edit Format View Help

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

Create/Modify Source Code

Saved on the disk

Source Code

.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
```

Compile Source Code
e.g., `javac Welcome.java`

If compile errors occur

Stored on the disk

Bytecode

.class

Run Bytecode
e.g., `java Welcome`

"Welcome to Java " is displayed on the console

```
Welcome to Java!
```

Result

If runtime errors or incorrect result

Compiling Java Source Code

You can port a source program to any machine with appropriate compilers. The source program must be recompiled, however, because the object program can only run on a specific machine. Nowadays computers are networked to work together. Java was designed to run object programs on any platform. 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.

(a)

(b)
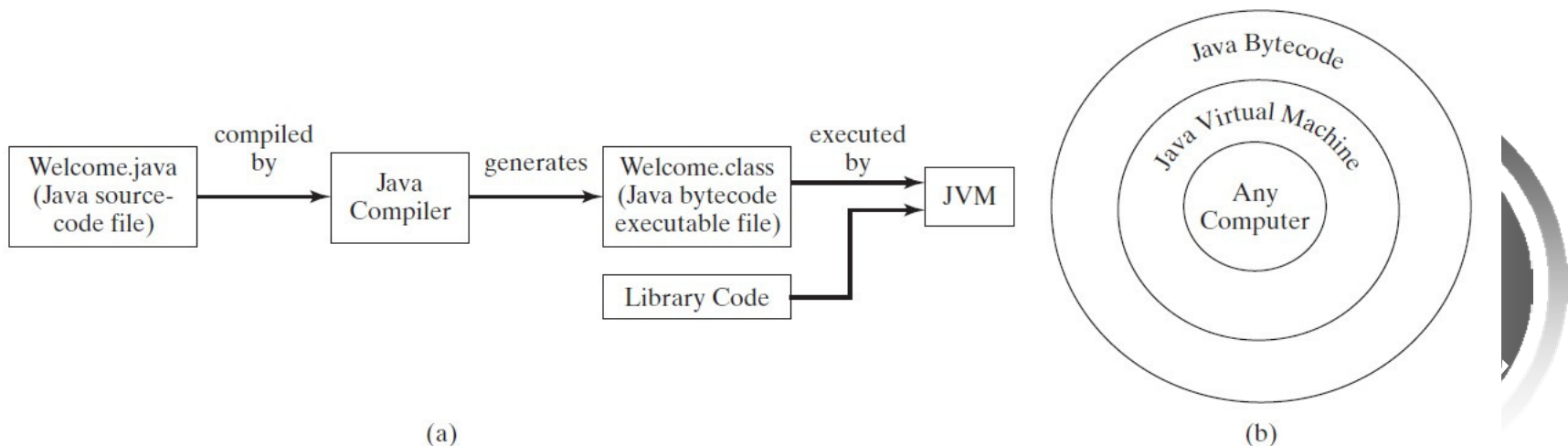
Trace a Program Execution

Enter main method

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

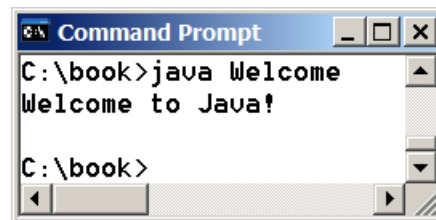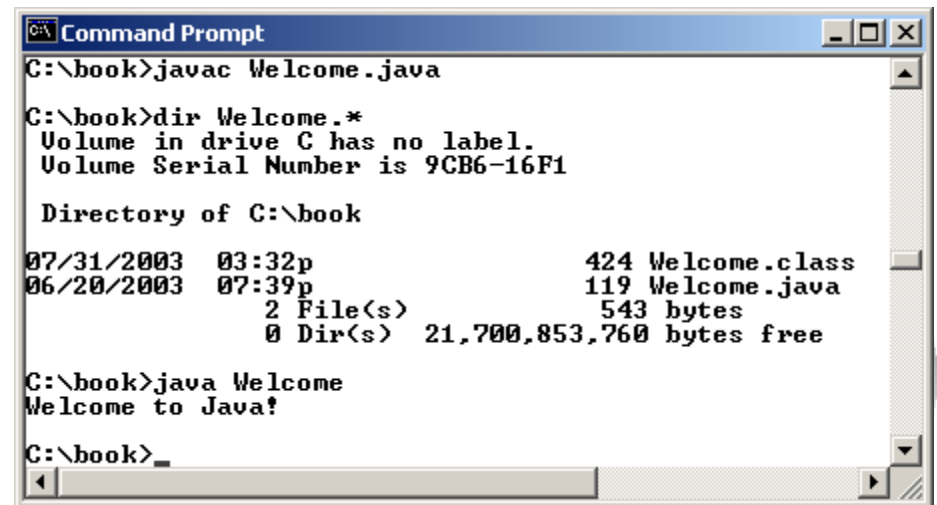# Trace a Program Execution

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

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

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

C:\book>
```

print a message to the console

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.

```java
// 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 in Listing 1.1 is a statement to display the greeting "Welcome to Java!".

```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.

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

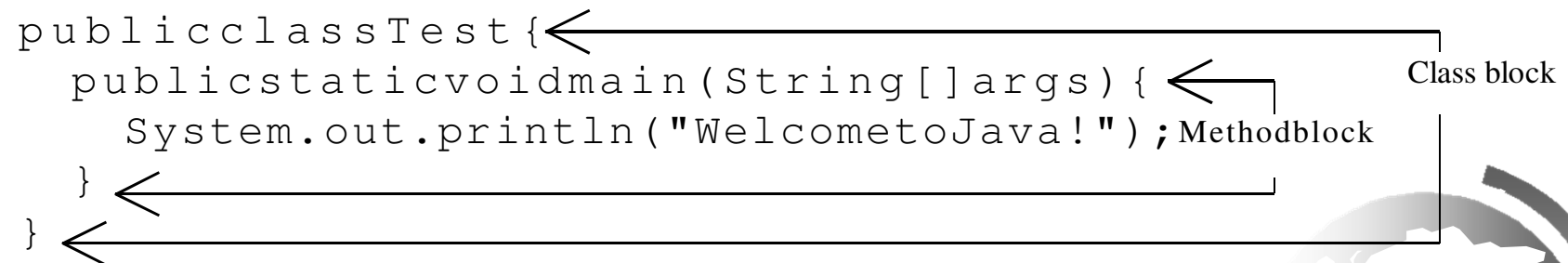| Reserved Words				
abstract	default	goto	package	synchronized
assert	do	if	private	this
boolean	double	implements	protected	throw
break	else	import	public	throws
byte	enum	instanceof	return	transient
case	extends	int	short	true
catch	false	interface	static	try
char	final	long	strictfp	void
class	finally	native	super	volatile
const	float	new	switch	while
continue	for	null		

Blocks

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

```
publicclassTest{
  publicstaticvoidmain(String[]args){
    System.out.println("WelcometoJava!");
  }
}
```

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!");
  }
}
```

(...)

```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

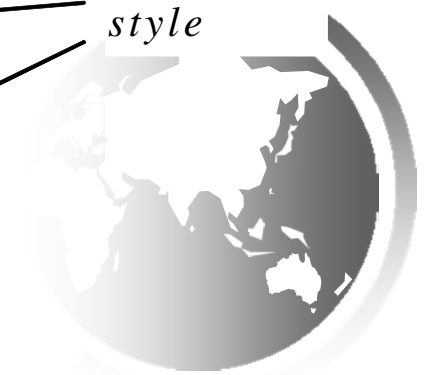 – Use blank line to separate segments of the code.

Block Styles

Use end-of-line style for braces.

Next-line style

```
publicclassTest
{
   publicstaticvoidmain(String[]args)
   {
     System.out.println("BlockStyles");
   }
}
```

End-of-line style

```
publicclassTest{
   publicstaticvoidmain(String[]args){
     System.out.println("BlockStyles");
   }
}
```

Programming Errors

☐ Syntax Errors

– Detected by the compiler

☐ Runtime Errors

– Causes the program to abort

☐ Logic Errors

– Produces incorrect result