

Chapter 1 Introduction to Computers, Programs, and Java



1

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.



2

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
```



3

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:

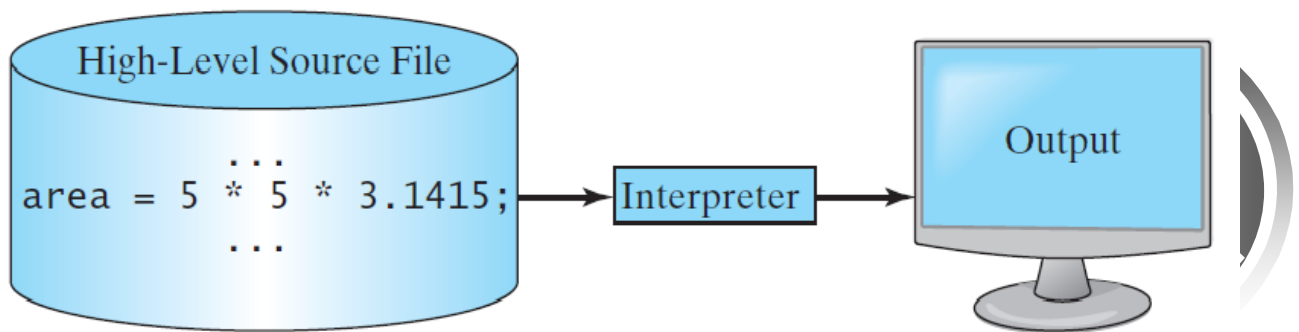
```
area = 5 * 5 * 3.1415;
```



4

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.



5

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.



6

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.



7

Java's History

James Gosling and Sun Microsystems

Oak

Java, May 20, 1995, Sun World

HotJava

- The first Java-enabled Web browser



8

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



9

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.



10

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

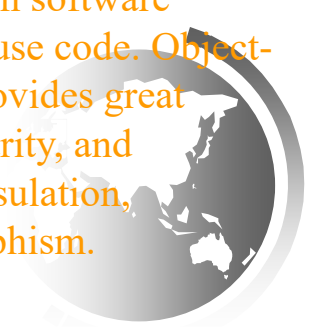
Java Is Multithreaded

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.



11

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.



12

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).



13

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.



14

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



15

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

Java Is Multithreaded

Java Is Dynamic

With a Java Virtual Machine (JVM), you can write one program that will run on any platform.



16

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.



17

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



18

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.



19

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.



20

JDK Versions

JDK 1.0 (1996)
JDK 1.1 (1997)
JDSE 1.2 (1998)
JDSE 1.3 (2000)
JDSE 1.4 (2002)
JDSE 5.0 (2004)
Java SE 6 (2006)
Java SE 7 (2011)
Java SE 8 (2014)
... Java SE 19 (2022)



21

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 Class uses **J2SE** to introduce Java programming.



22

Popular Java IDEs

NetBeans

Eclipse

IntelliJ IDEA



23

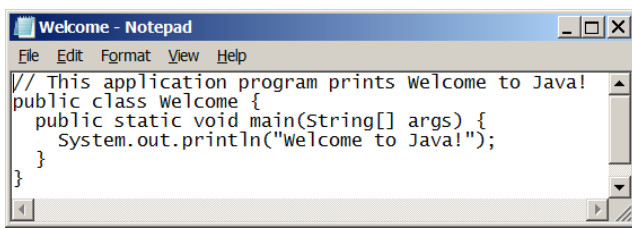
A Simple Java Program

Listing 1.1

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



24



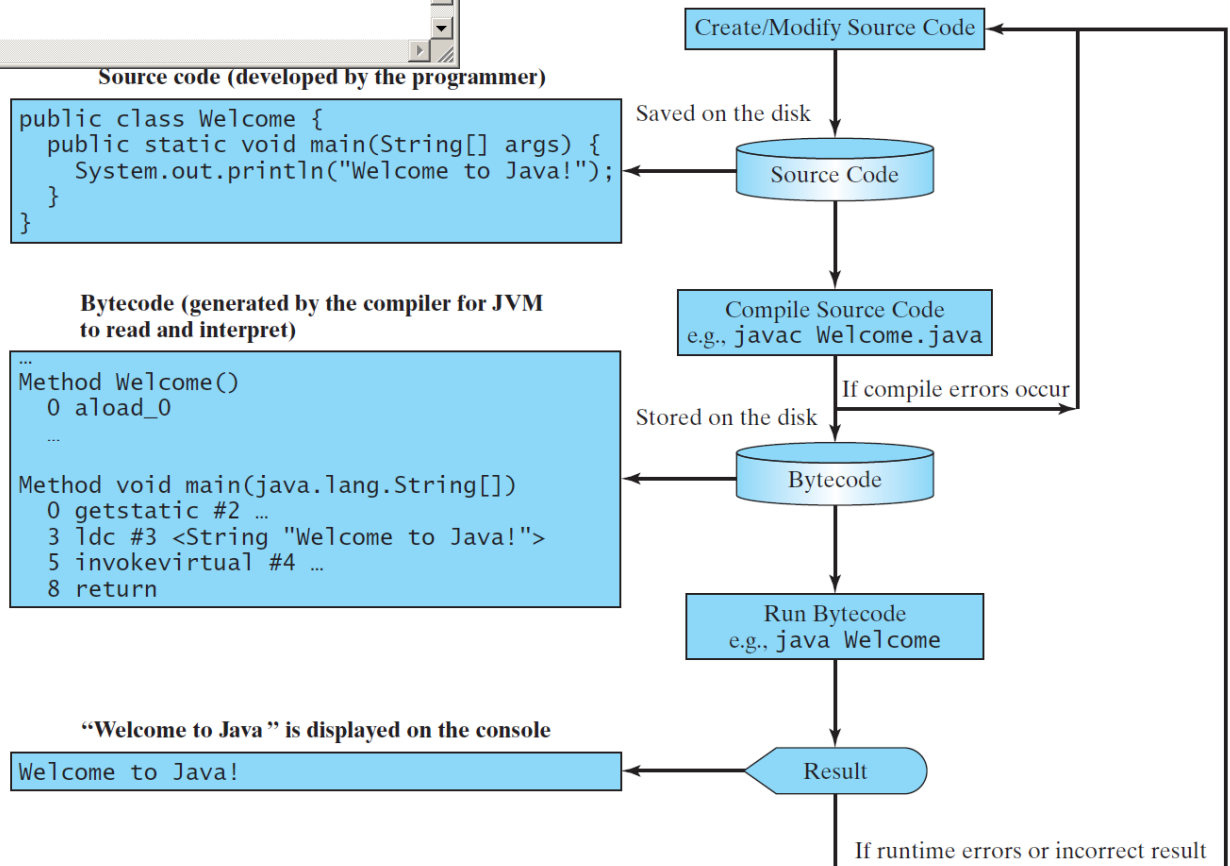
Source code (developed by the programmer)

```
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  
...
```

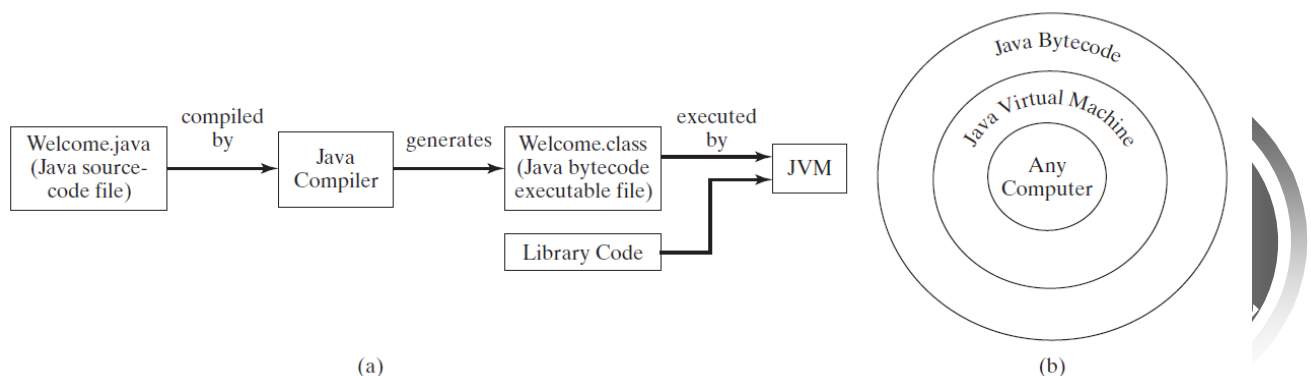
Creating, Compiling, and Running Programs



25

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 software that interprets Java bytecode.



26

Anatomy of a Java Program

Class name

Main method

Statements

Statement terminator

Reserved words

Comments

Blocks



27

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!");
    }
}
```




28

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!");
    }
}
```




29

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!".

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




30

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!");
    }
}
```




31

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!");
    }
}
```



32


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



33

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.



34

{ ... }


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



35

(...)


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



36

·
;


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



37

// ...


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



38

" ... "

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



39

Programming Style and Documentation

Appropriate Comments

Naming Conventions

Proper Indentation and Spacing Lines

Block Styles



40

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.



41

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`.



42

Proper Indentation and Spacing

Indentation

- Indent two spaces.

Spacing

- Use blank line to separate segments of the code.



43

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");
    }
}
```



44

Programming Errors

Syntax Errors

- Detected by the compiler

Runtime Errors

- Causes the program to abort

Logic Errors

- Produces incorrect result



45

Syntax Errors

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

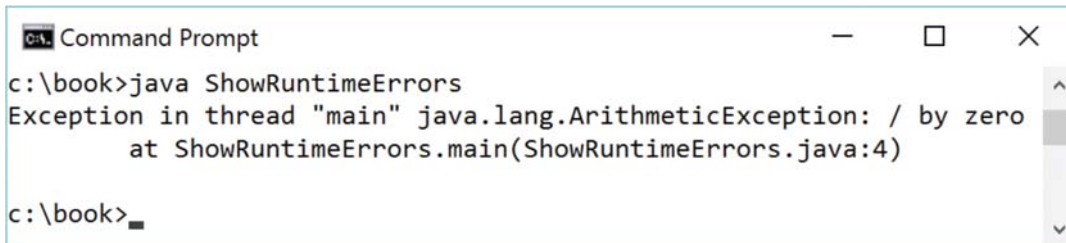
```
Command Prompt  
c:\book>javac ShowSyntaxErrors.java  
ShowSyntaxErrors.java:3: error: invalid method declaration; return type required  
    public static main(String[] args) {  
            ^  
ShowSyntaxErrors.java:4: error: unclosed string literal  
        System.out.println("Welcome to Java);  
                        ^  
ShowSyntaxErrors.java:4: error: ';' expected  
        System.out.println("Welcome to Java);  
                        ^  
ShowSyntaxErrors.java:6: error: reached end of file while parsing  
    }  
    ^  
4 errors  
c:\book>
```



46

Runtime Errors

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



The screenshot shows a Windows Command Prompt window titled "Command Prompt". The command entered is `c:\book>java ShowRuntimeErrors`. The output is an exception message: `Exception in thread "main" java.lang.ArithmeticException: / by zero` followed by the stack trace `at ShowRuntimeErrors.main(ShowRuntimeErrors.java:4)`. The prompt is now `c:\book>` with a cursor.



47

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);  
    }  
}
```

Celsius 35 is Fahrenheit degree 67 (X)



48