

Chapter 1 Introduction to Programs, and Java



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

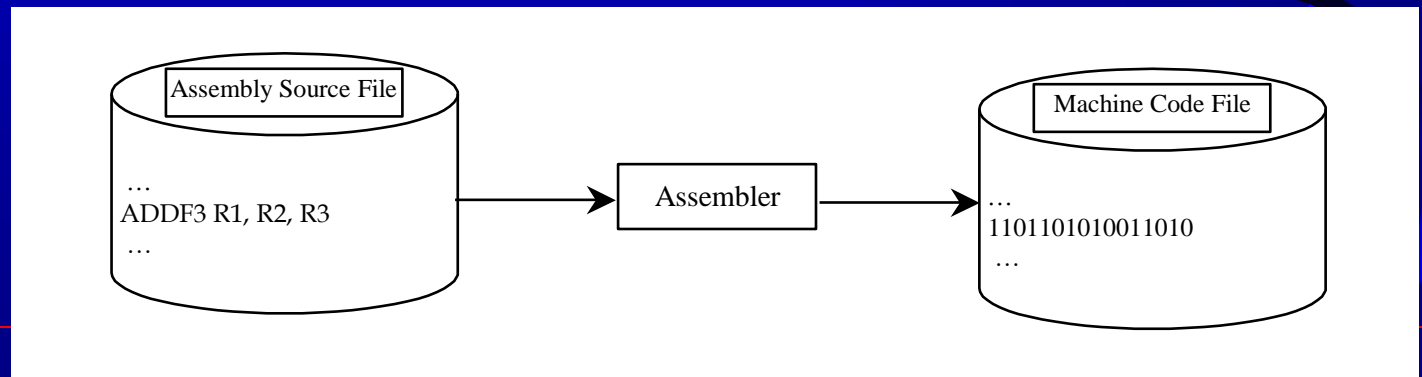


Programming Languages

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:

```
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;
```



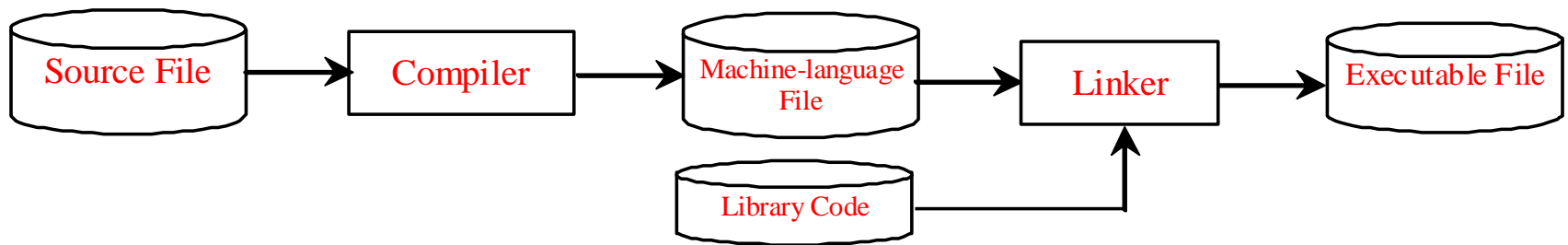
Popular High-Level Languages

- ☞ COBOL (COmmon Business Oriented Language)
- ☞ FORTRAN (FORmula TRANslation)
- ☞ BASIC (Beginner All-purpose Symbolic Instructional Code)
- ☞ Pascal (named for Blaise Pascal)
- ☞ Ada (named for Ada Lovelace)
- ☞ C (whose developer designed B first)
- ☞ Visual Basic (Basic-like visual language developed by Microsoft)
- ☞ Delphi (Pascal-like visual language developed by Borland)
- ☞ C++ (an object-oriented language, based on C)
- ☞ C# (a Java-like language developed by Microsoft)
- ☞ Java (We use it for the course)



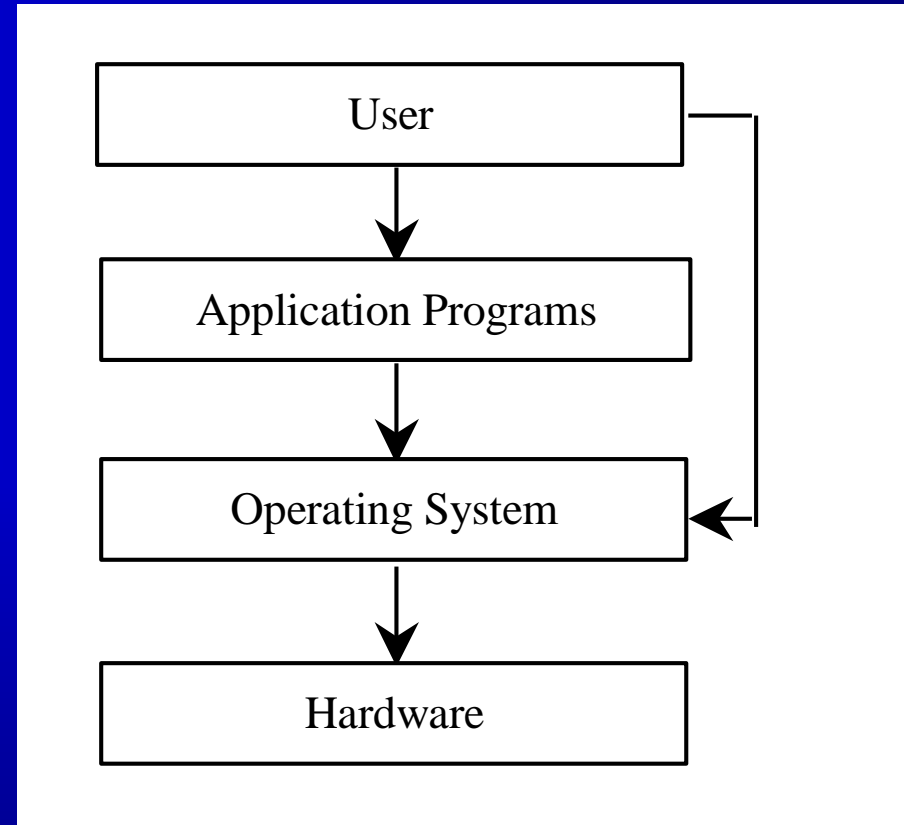
Compiling Source Code

A program written in a high-level language is called a *source program*. Since a computer cannot understand a source program. Program called a *compiler* is used to translate the source program into a machine language program called an *object program*. The object program is often then linked with other supporting library code before the object can be executed on the machine.



Operating Systems

The *operating system* (OS) is a program that manages and controls a computer's activities. You are probably using Windows 98, NT, 2000, XP, or ME. Windows is currently the most popular PC operating system. Application programs such as an Internet browser and a word processor cannot run without an 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. 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.

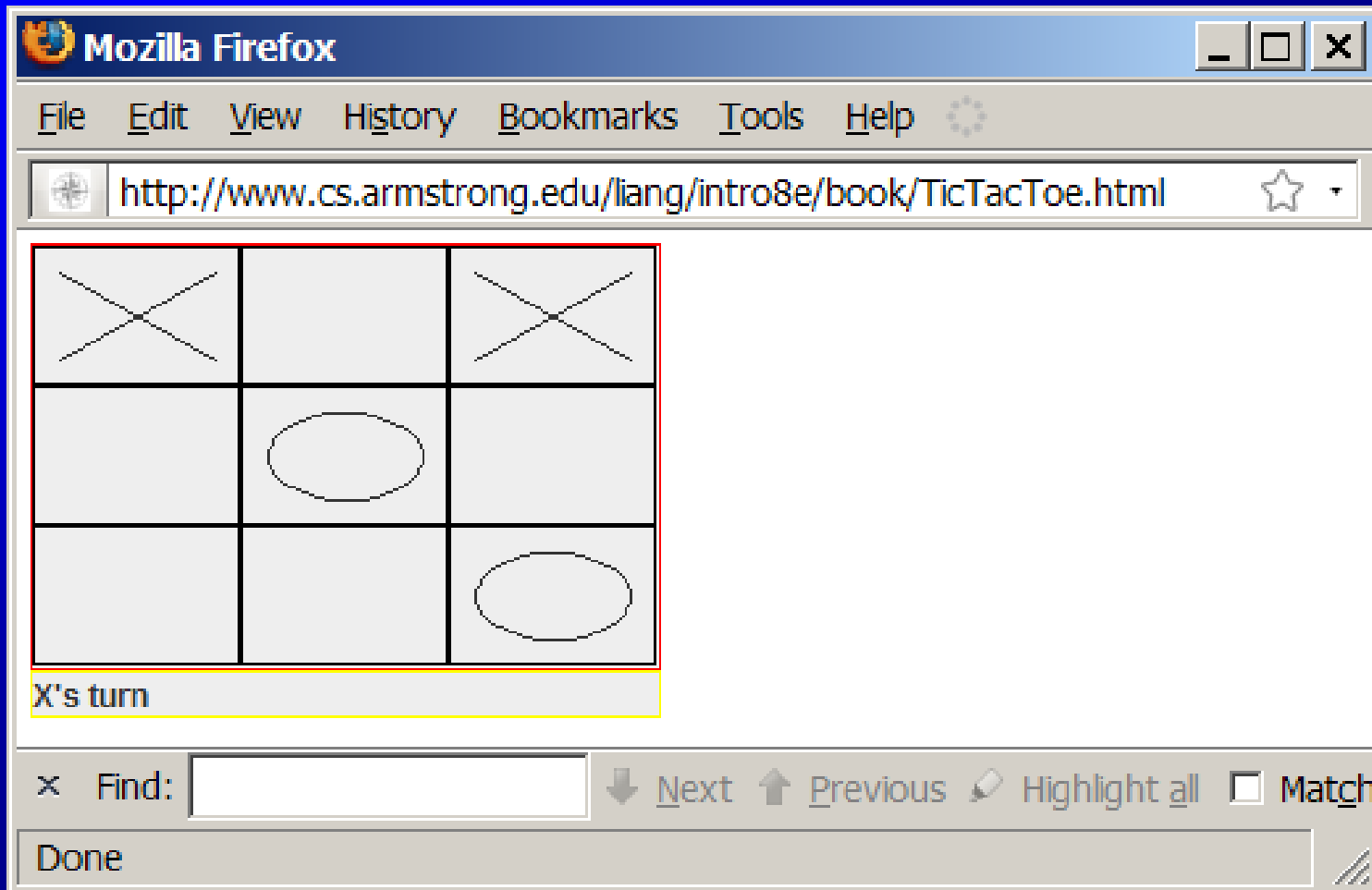


Java, Web, and Beyond

- ☞ Java can be used to develop Web applications.
- ☞ Java Applets
- ☞ Java Web Applications
- ☞ Java can also be used to develop applications for hand-held devices such as Palm and cell phones



Examples of Java's Versatility (Applets)



PDA and Cell Phone



Java's History

☞ James Gosling and Sun Microsystems

☞ Oak

☞ Java, May 20, 1995, Sun World

☞ HotJava

– The first Java-enabled Web browser

☞ Early History Website:

<http://java.sun.com/features/1998/05/birthday.html>

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/intro8e/JavaCharacteristics.pdf

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.

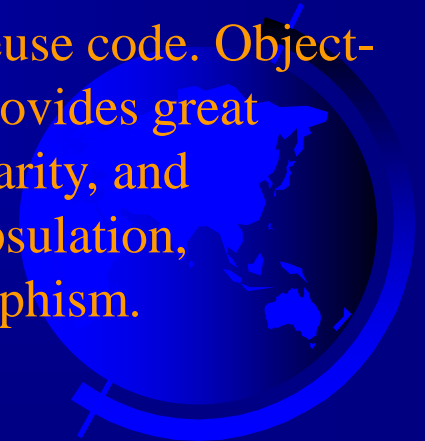


Characteristics of Java

- ➡ 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 Is Architecture-Neutral
- ➡ Java Is Portable
- ➡ Java's Performance
- ➡ Java Is Multithreaded
- ➡ Java Is Dynamic

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



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

Write once, run anywhere

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



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

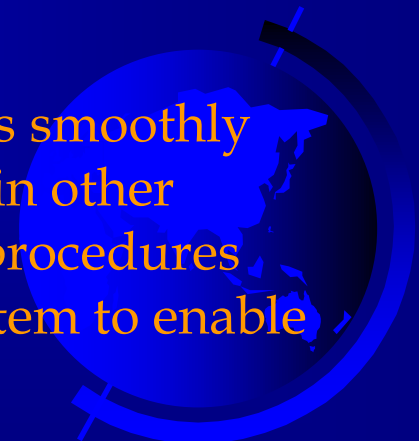
Java's performance Because Java is architecture neutral, Java programs are portable. They can be run on any platform without being recompiled.



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

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

- ➡ 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 (possibly 2010) a. k. a. JDK 7 or Java 7



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 and Java ServerPages.

☞ 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 Open Source by Sun
- ☞ Eclipse Open Source by IBM
- ☞ Jcreator LE Open Source by Sun for Learners

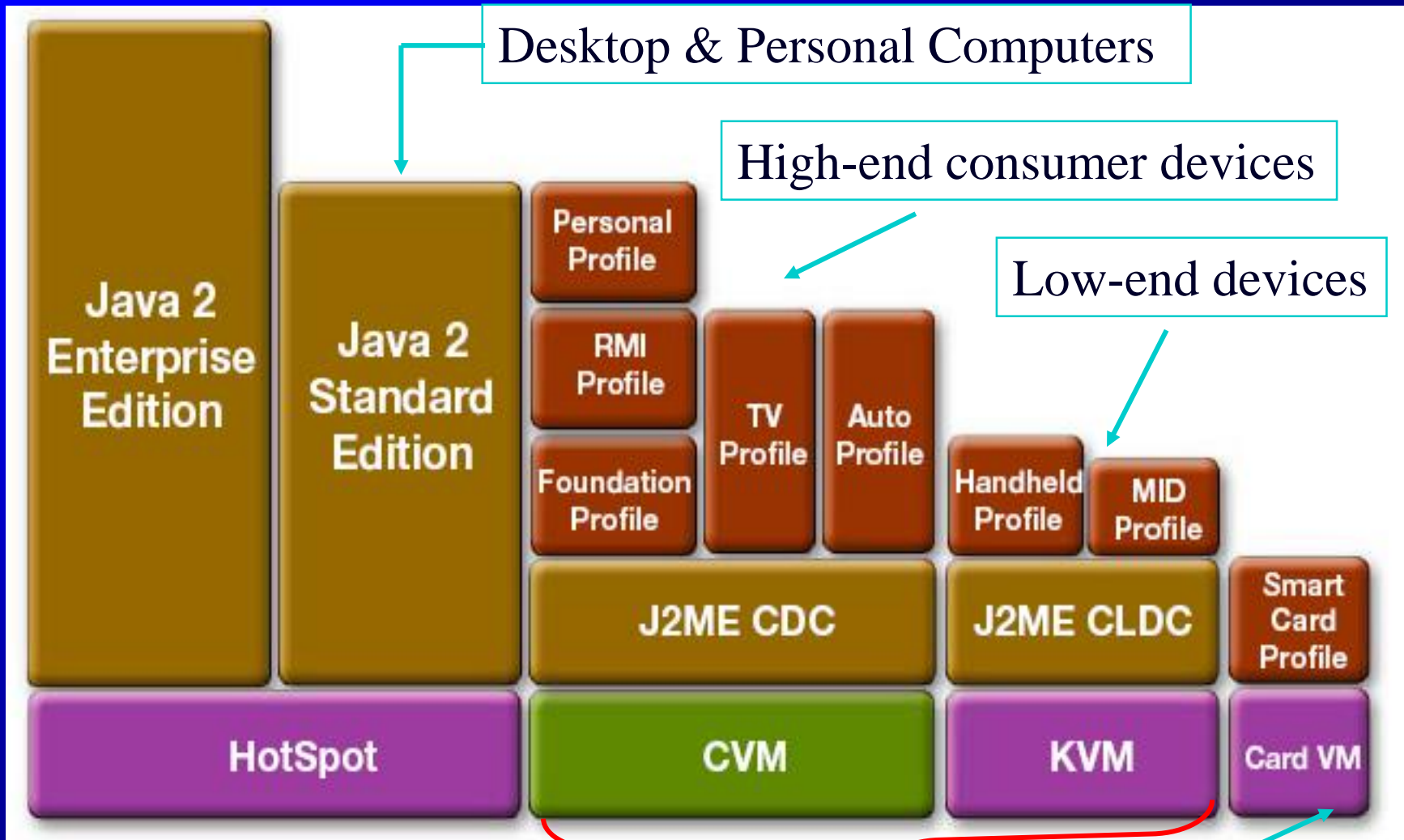


servers & enterprise computers

Desktop & Personal Computers

High-end consumer devices

Low-end devices



J2ME

smartcards

Java 2 Platform editions and their target markets

A Simple Java Program

Listing 1.1

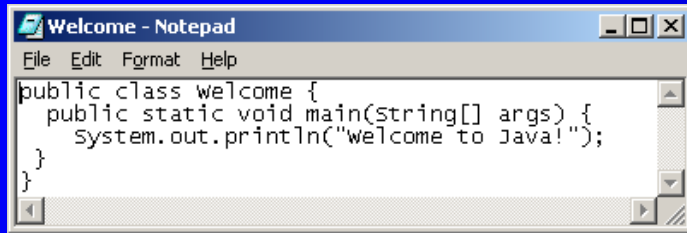
```
//This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

Welcome

Run



Creating, Compiling, and Running Programs



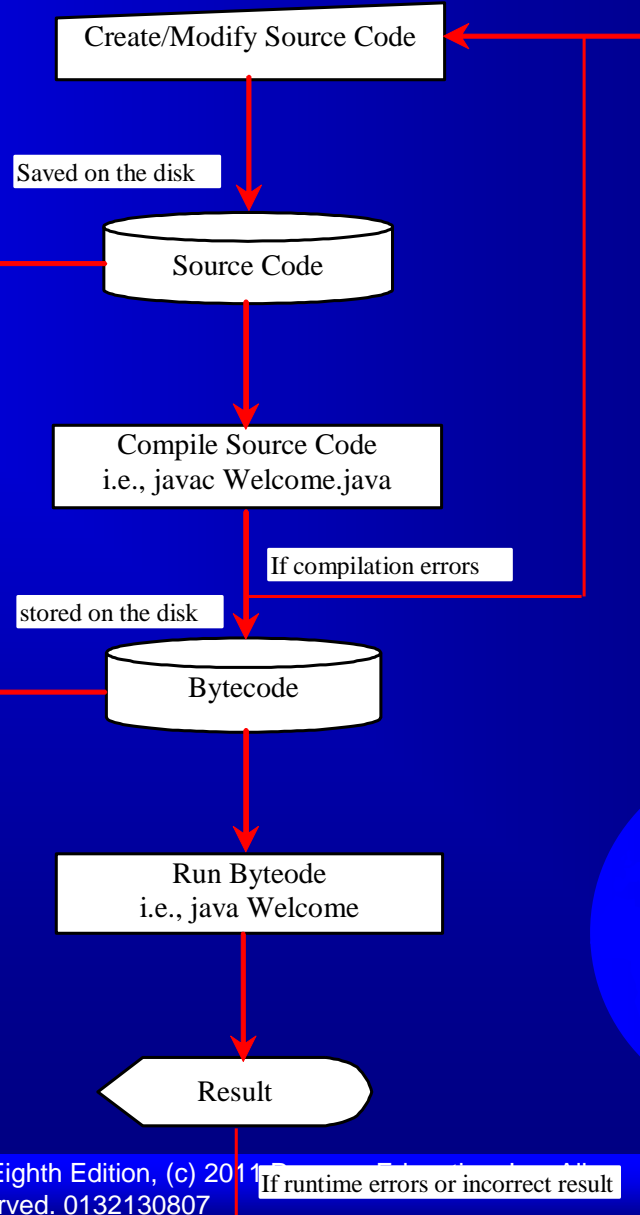
```
public class welcome {  
    public static void main(String[] args) {  
        System.out.println("welcome to Java!");  
    }  
}
```

Source code (developed by the programmer)

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

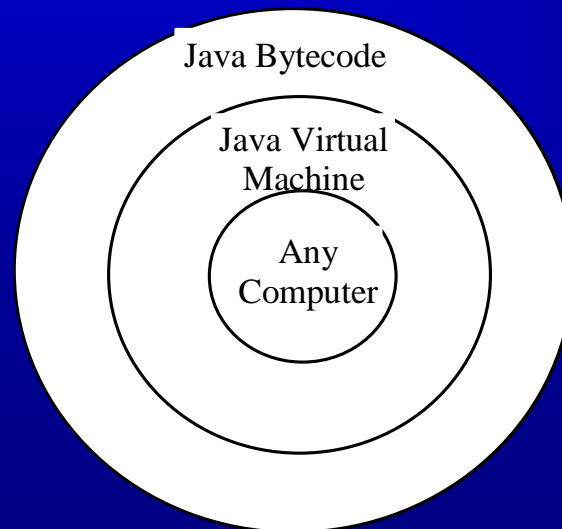
Byte code (generated by the compiler for JVM to read and interpret, not for you to understand)

```
...  
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
```



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.



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!");  
    }  
}
```

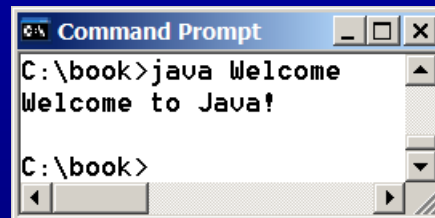
Trace a Program Execution

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!");  
    }  
}
```



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

print a message to the
console

Anatomy of a Java Program

- ➡ Comments
- ➡ Reserved words
- ➡ Modifiers
- ➡ Statements
- ➡ Blocks
- ➡ Classes
- ➡ Methods
- ➡ The main method



Comments

Three types of comments in Java.

Line comment: A line comment is preceded by two slashes (//) in a line.

Paragraph comment: A paragraph comment is enclosed between /* and */ in one or multiple lines.

javadoc comment: javadoc comments begin with /** and end with */. They are used for documenting classes, data, and methods. They can be extracted into an HTML file using JDK's javadoc command.

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. Other reserved words in Listing 1.1 are `public`, `static`, and `void`. Their use will be introduced later in the book.



Modifiers

Java uses certain reserved words called modifiers that specify the properties of the data, methods, and classes and how they can be used. Examples of modifiers are public and static. Other modifiers are private, final, abstract, and protected. A public datum, method, or class can be accessed by other programs. A private datum or method cannot be accessed by other programs. Modifiers will be discussed later.”



Statements

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



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



Classes

The class is the essential Java construct. A class is a template or blueprint for objects. To program in Java, you must understand classes and be able to write and use them. The mystery of the class will continue to be unveiled throughout this book. For now, though, understand that a program is defined by using one or more classes.



Methods

What is `System.out.println`? It is a method: a collection of statements that performs a sequence of operations to display a message on the console. It can be used even without fully understanding the details of how it works. It is used by invoking a statement with a string argument. The string argument is enclosed within parentheses. In this case, the argument is "Welcome to Java!" You can call the same `println` method with a different argument to print a different message.



main Method

The main method provides the control of program flow. The Java interpreter executes the application by invoking the main method.

The main method looks like this:

```
public static void main(String[] args) {  
    // Statements;  
}
```

