HZ BOOKS
华章教育

全美经典
学习指导系列

Java编程习题与解答

(英文版)

Programming with Java

Master today's hottest programming language

Create dynamic animations, simulations, and interactive Web pages

Learn by example

Over 200 fully-solved problems

The perfect aid for better grades

全球销售超过3000万册!

(美) John R. Hubbard 著

Mc Graw Hill

机械工业出版社
China Machine Press

全美经典
学习指导系列

Java 编程习题与解答

（英文版）

PROGRAMMING WITH JAVA

（美）John R. Hubbard 著

机械工业出版社
China Machine Press

本书版权登记号：图字：01-2002-2191

图书在版编目（CIP）数据

Java 编程习题与解答/（美）哈伯特（Hubbard, J R）著 -北京 机械工业出版社，2002.8

（全美经典学习指导系列）

书名原文 Programmıng Wıth Java

ISBN 7-111-10582-6

I J　II 哈　III JAVA语言-程序设计-习题-英文 IV TP312-44

中国版本图书馆CIP数据核字（2002）第051852号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑 华 章

北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行

2002年8月第1版第1次印刷

787mm × 1092mm 1/16 16 75印张

印数 0 001- 3000册

定价 28 00元

Preface

Like all Schaum's Outline Series books, this is intended to be used primarily for self study, preferably in conjunction with a regular course in the fundamentals of computer science using the Java programming language.

The book includes over 200 examples and problems. The author firmly believes that programming is learned best by practice, following a well-constructed collection of examples with complete explanations. This book is designed to provide that support.

Source code for all the examples, solved problems, and supplementary problems in this book may be downloaded from the author's Web page:

`http://www.richmond.edu/~hubbard/`

This site will also contain any corrections and addenda for the book.

I wish to thank all my friends, colleagues, students, and the McGraw-Hill staff who have helped me with the critical review of this manuscript, including Eric Ciampa, Andrew Somers, Michael Somers, and Maureen Walker. Special thanks to Anita Hubbard for her advice, encouragement, and supply of creative problems for this book.

JOHN R. HUBBARD
Richmond, Virginia

Contents

Chapter 1 Getting Started **1**

1.1 THE JAVA PROGRAMMING LANGUAGE 1
1.2 DOWNLOADING THE JAVA DEVELOPMENT KIT 3
1.3 INSTALLING THE JDK 5
1.4 CREATING AND RUNNING A PROGRAM IN MICROSOFT WINDOWS 6
1.5 TROUBLESHOOTING 9
1.6 ANALYSIS OF THE **HelloWorld** PROGRAM 11
1.7 COMMENTS 12
1.8 PROGRAMS WITH INPUT 14
1.9 NUMERIC INPUT 16
1.10 VARIABLES AND OBJECTS 19
1.11 ARITHMETIC AND ASSIGNMENT OPERATORS 21

Chapter 2 Strings **27**

2.1 THE **String** CLASS 27
2.2 SUBSTRINGS 28
2.3 CHANGING CASE 29
2.4 CONCATENATION 29
2.5 LOCATING A CHARACTER WITHIN A STRING 30
2.6 REPLACING CHARACTERS IN A STRING 31
2.7 REPRESENTING A PRIMITIVE VALUE IN A STRING 31
2.8 SUMMARY OF THE **String** CLASS METHODS 33
2.9 THE **StringBuffer** CLASS 34
2.10 SUMMARY OF THE **StringBuffer** CLASS METHODS 39

Chapter 3 Selection **43**

3.1 THE **if** STATEMENT 43
3.2 THE **if...else** STATEMENT 44
3.3 THE **if...else if...** STATEMENT COMBINATION 45
3.4 NESTED CONDITIONALS 45
3.5 COMPOUND STATEMENTS 48
3.6 OPERATORS 48
3.7 ORDER OF EVALUATION 50
3.8 **boolean** VARIABLES 51
3.9 THE CONDITIONAL OPERATOR 54
3.10 ASSIGNMENT OPERATORS 55
3.11 THE **switch** STATEMENT 55

Chapter 4 Iteration **65**

4.1 THE **for** STATEMENT 65
4.2 THE **while** STATEMENT 68
4.3 SOME NUMBER CRUNCHING 70
4.4 THE **do...while** STATEMENT 73
4.5 MORE NUMBER CRUNCHING 75
4.6 NESTED LOOPS 78

Chapter 5 **Methods** **89**

5.1 SIMPLE EXAMPLES 89
5.2 LOCAL VARIABLES 90
5.3 METHODS OFTEN INVOKE OTHER METHODS 92
5.4 METHODS THAT INVOKE THEMSELVES 93
5.5 **`boolean`** METHODS 95
5.6 **`void`** METHODS 95
5.7 OVERLOADING 96

Chapter 6 **Classes** **110**

6.1 CLASSES 110
6.2 DECLARATIONS 113
6.3 MODIFIERS 116
6.4 CONSTRUCTORS 118
6.5 COPY CONSTRUCTORS 121
6.6 DEFAULT CONSTRUCTORS 122
6.7 CLASS INVARIANTS 124
6.8 IDENTITY, EQUALITY, AND EQUIVALENCE 127
6.9 MORE CLASS INVARIANTS 129
6.10 WRAPPER CLASSES 133

Chapter 7 **Composition and Inheritance** **148**

7.1 COMPOSITION 148
7.2 RECURSIVE CLASSES 152
7.3 INHERITANCE 157
7.4 OVERRIDING FIELDS AND METHODS 159
7.5 THE **`super`** KEYWORD 161
7.6 INHERITANCE VERSUS COMPOSITION 163
7.7 CLASS HIERARCHIES 163
7.8 THE **`Object`** CLASS 167
7.9 THE JAVA CLASS HIERARCHY 168
7.10 THE **`clone()`** AND **`equals()`** METHODS 168

Chapter 8 **Arrays and Vectors** **188**

8.1 CHARACTER ARRAYS 188
8.2 PROPERTIES OF ARRAYS IN JAVA 191
8.3 COPYING AN ARRAY 193
8.4 THE **`Vector`** CLASS 195
8.5 THE SIZE AND CAPACITY OF A **`Vector`** OBJECT 198
8.6 CHANGES TO THE **`Vector`** CLASS IN JAVA 1.2 200
8.7 TWO-DIMENSIONAL ARRAYS 201

Chapter 9 Graphics **209**

9.1 THE AWT LIBRARIES 209
9.2 THE **Frame** CLASS 209
9.3 THE **Color** CLASS 212
9.4 COMPONENTS 213
9.5 THE **Button** CLASS 214
9.6 MANAGING LAYOUTS 215
9.7 EVENT-DRIVEN PROGRAMMING 217
9.8 THE **TextField** CLASS 220

Chapter 10 Applets, Threads, and Exceptions **224**

10.1 APPLETS 224
10.2 THE **Applet** CLASS 225
10.3 THE **Thread** CLASS 227
10.4 EXCEPTIONS 230

Appendix A Acronyms **234**

Appendix B The JBuilder IDE **235**

Appendix C The CodeWarrior IDE **238**

Appendix D Computer Numbers **239**

D.1 THE FINITE AND THE INFINITE 239
D.2 NUMBER TYPES 239
D.3 RANGES OF INTEGER TYPES 239
D.4 INTEGER OVERFLOW 239
D.5 RANGES OF FLOATING-POINT TYPES 241
D.6 THE **INFINITY** AND **NaN** CONSTANTS 241

Appendix E Unicode **244**

Appendix F References **246**

Index **248**

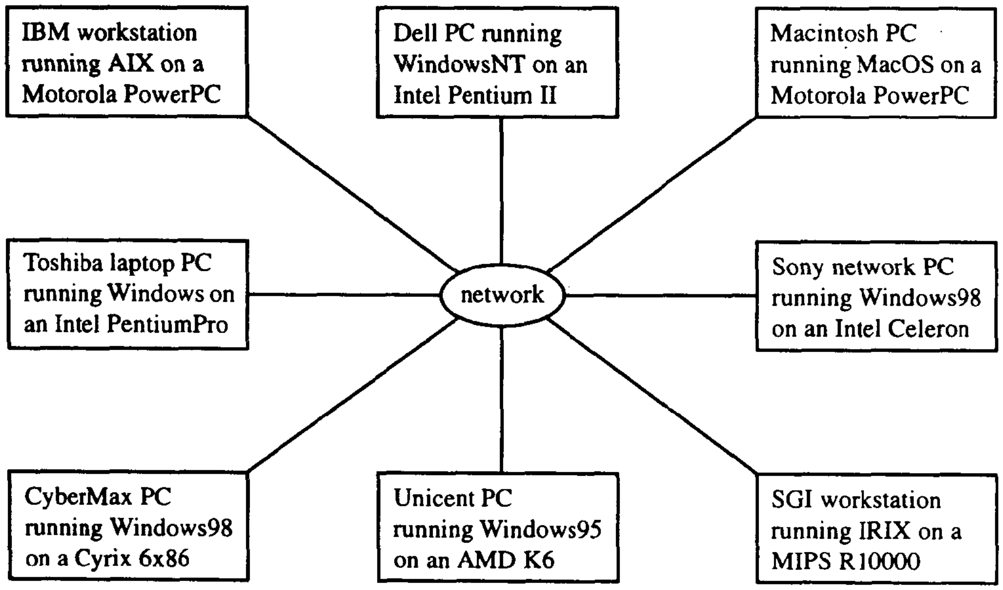
Chapter 1

Getting Started

1.1 THE JAVA PROGRAMMING LANGUAGE

The Java programming language was developed by James Gosling at Sun Microsystems in 1991. Its name is a slang term for coffee. When the World Wide Web appeared on the Internet in 1993, the language was enhanced to facilitate programming on the web. Since then it has become one of the most popular languages, especially for network programming.

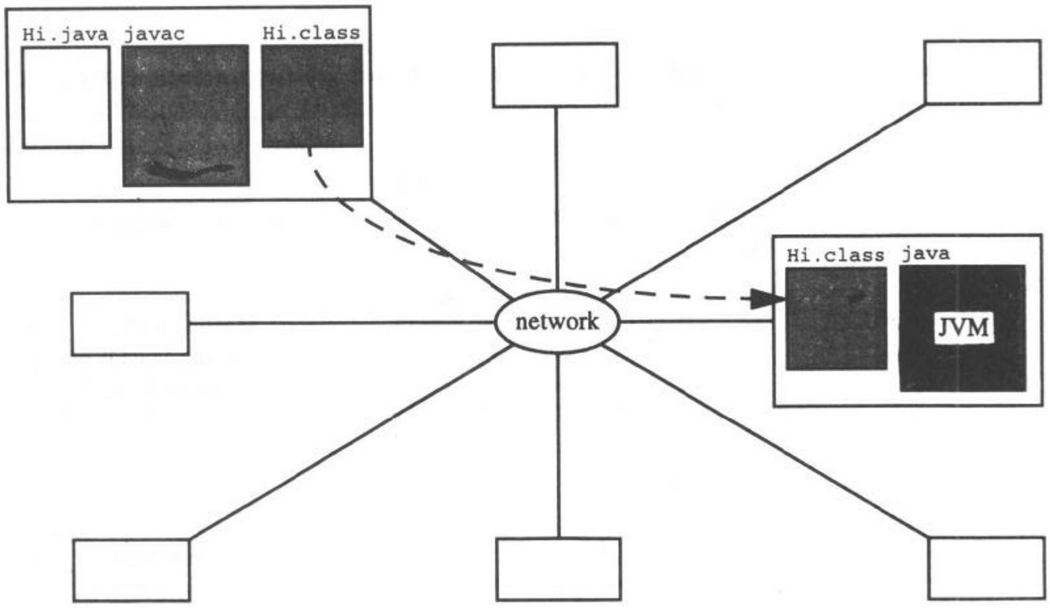
To see why Java is the language of choice among network programmers, imagine a network of different computers like this:


IBM workstation running AIX on a Motorola PowerPC
Dell PC running WindowsNT on an Intel Pentium II
Macintosh PC running MacOS on a Motorola PowerPC
Toshiba laptop PC running Windows on an Intel PentiumPro
network
Sony network PC running Windows98 on an Intel Celeron
CyberMax PC running Windows98 on a Cyrix 6x86
Unicent PC running Windows95 on an AMD K6
SGI workstation running IRIX on a MIPS R10000

These might be eight computers in the same room connected within a local area network, or they could be in eight different cities on four continents connected by the Internet. The point is that they are running different operating systems (AIX, WindowsNT, *etc.*) on different processors (PowerPC, Pentium II, *etc.*). Suppose that you want to write a program on the IBM workstation that can be run on all eight computers.

Before a computer program can be run, it has to be translated into the machine language that the computer's processor understands. In programming languages such as Pascal and C++, this translation is done by a *compiler*, and the resulting machine language version of the program is called the *executable image*. But different processors have different machine languages. So, for example, an executable image produced on the IBM workstation would not run on any of the other computers in the network shown above. To have his or her program run on all the computers in the network, the programmer would have to compile it separately on each one!

To solve this problem, Java provides both a compiler and a software system called the Java Virtual Machine (JVM) for each computer system. The Java compiler, `javac`, translates Java source code into an intermediate level language, called ***bytecodes***. Like the source code itself, bytecodes are independent of the type of computer system. The same bytecode file can be used by any computer. When one computer wants to run a Java program written on another computer, it downloads that program's bytecode file and delivers it to its own JVM. The JVM then translates the bytecodes into its own system's machine language and runs the result.



This picture represents the same network as before. It shows three files on the IBM computer: a Java source code file named `Hi.java`, the `javac` compiler, and the Java bytecode file named `Hi.class`. That bytecode file was produced by the compiler when the programmer executed the command

```
    javac Hi.class
```

on the IBM computer. Later, a user at the Sony computer on the right clicked on a Web page that includes instructions to run the `Hi.class` program. In response, the Sony computer downloaded the `Hi.class` bytecode file from the IBM computer and ran its own local JVM on it. The JVM on the Sony computer knows how to translate the bytecode into its own processor's machine language so it can be executed there. All the previous work done on the IBM computer is completely independent of the Sony computer. In fact, the `Hi.class` bytecode file could have been produced long before the Sony computer or its Intel Celeron processor were ever invented!

Most Web browsers (Netscape's Communicator, Microsoft's Internet Explorer, *etc.*) come bundled with the JVM. So when you load a web page that includes instructions to run a Java program, the browser automatically downloads the bytecode file and runs the JVM on it. All you see are the results on your web page: animated images, data entry forms, buttons, scroll panes, check boxes, *etc.*

The JVM system is an *interpreter*. That means that it translates and runs each bytecode instruction separately, whenever it is needed by the complete program. For some programs, this can be quite slow. As an alternative, Java also provides local compilers for each system that will compile a bytecode file into an executable image for faster running. Java calls these compilers "Just-In-Time" (JIT) compilers. They come bundled with some web browsers (*e.g.*, Netscape).

1.2 DOWNLOADING THE JAVA DEVELOPMENT KIT

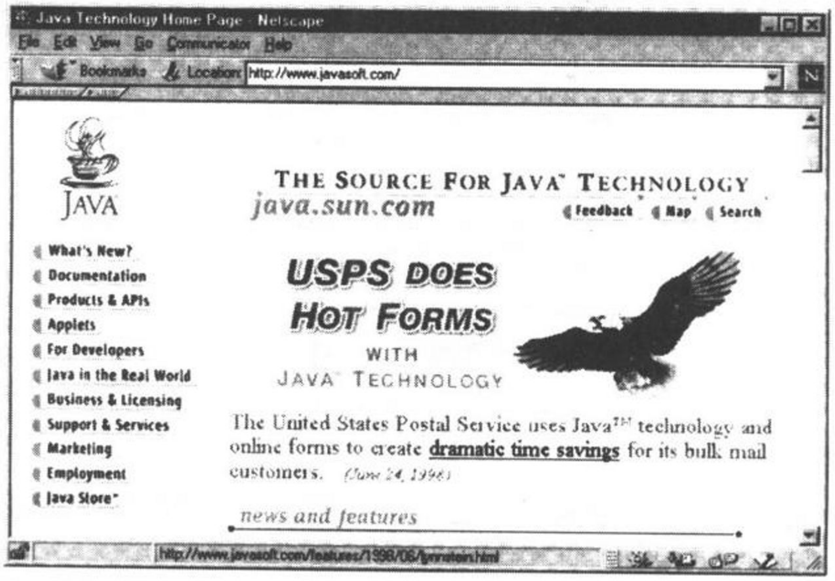
The process of designing, coding, testing, debugging, documenting, maintaining, and upgrading computer programs is called *software development*. An Integrated Development Environment (IDE) is a collection of integrated programs that facilitates software development. If you have access to an IDE (*e.g.*, Metrowerks's CodeWarrior, Enprise's JBuilder, Microsoft's Visual J++, Symantec's Visual Cafe, or IBM's Visual Age) skip to Section 1.6 on page 11.

The Java Development Kit (JDK) is a collection of programs to help developers compile, run, and debug Java programs. It is not as good as an IDE, but it is quite adequate for developing Java programs. Sun Microsystems provides it free of charge. This section describes how to download it from Sun's `javasoft` website.

To download the JDK to your computer, open your web browser (*e.g.*, Netscape Navigator) and enter the following URL in your browser's **Location** or **Address** field:

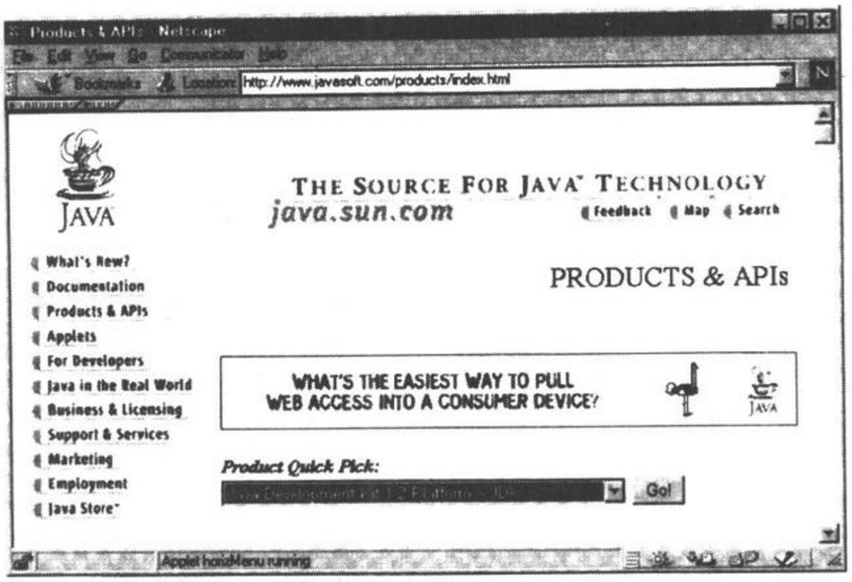
```
http://www.javasoft.com
```

This brings up the Java home page which should look something like this:


Java Technology Home Page - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://www.javasoft.com/
JAVA
THE SOURCE FOR JAVA TECHNOLOGY
java.sun.com
Feedback Map Search
What's New?
Documentation
Products & APIs
Applets
For Developers
Java in the Real World
Business & Licensing
Support & Services
Marketing
Employment
Java Store
USPS DOES HOT FORMS WITH JAVA TECHNOLOGY
news and features
http://www.javasoft.com/features/1998/06/grinstein.html

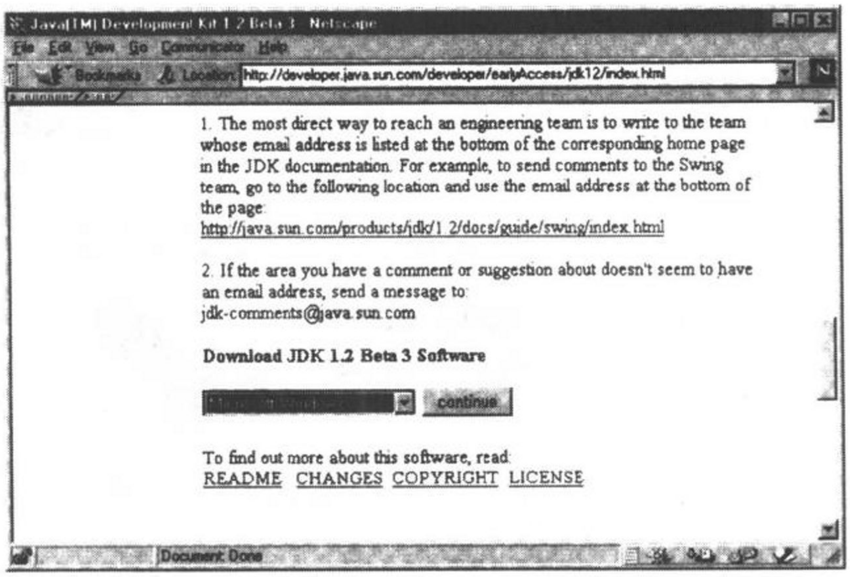
The United States Postal Service uses Java™ technology and online forms to create dramatic time savings for its bulk mail customers. (June 24, 1998)

Click on the link labeled **Products & APIs** (the third item on the left side of the page shown here). This brings up the PRODUCTS & APIs page. Use the pull-down menu labelled ***Product Quick Pick*** to select the most recent version of the Java Development Kit. In the window shown below, that was **Java Development Kit 1.2 Platform – JDK.**



Then click on the Go! button and follow the directions given on the next page that comes up to download the JDK. You may be asked to become a member of the Java Developer Connection, requiring you to submit a User ID and Password that you select yourself.

Eventually, you should get to another pull-down menu labelled **Download JDK**:


Java(TM) Development Kit 1.2 Beta 3 - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://developer.java.sun.com/developer/earlyAccess/jdk12/index.html

1. The most direct way to reach an engineering team is to write to the team whose email address is listed at the bottom of the corresponding home page in the JDK documentation. For example, to send comments to the Swing team, go to the following location and use the email address at the bottom of the page:
http://java.sun.com/products/jdk/1.2/docs/guide/swing/index.html

2. If the area you have a comment or suggestion about doesn't seem to have an email address, send a message to:
jdk-comments@java.sun.com

Download JDK 1.2 Beta 3 Software

To find out more about this software, read:
README CHANGES COPYRIGHT LICENSE

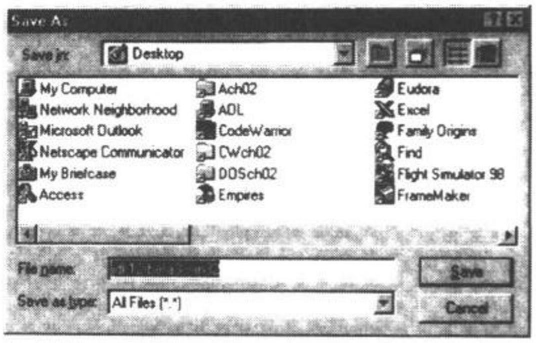
Select your computer's operating system and click on the continue button. You will then be asked to accept Sun's License & Export agreement.

When you finally get to the actual download button, it will probably be labelled with the name of the executable file that will downloaded, something like `jdk12-beta3-win32.exe`. Click on that button to begin the download.

If you are running Microsoft Windows, the system will bring up a panel like this:

Navigate up to the Desktop (as shown here) so that the file will appear there when it has finished downloading. Then press the Save button.

The JDK download is a large file. The one shown here (beta Release 1.2) is over 15 megabytes. So it may take well over an hour to download, depending upon the current traffic level at your site on the web.

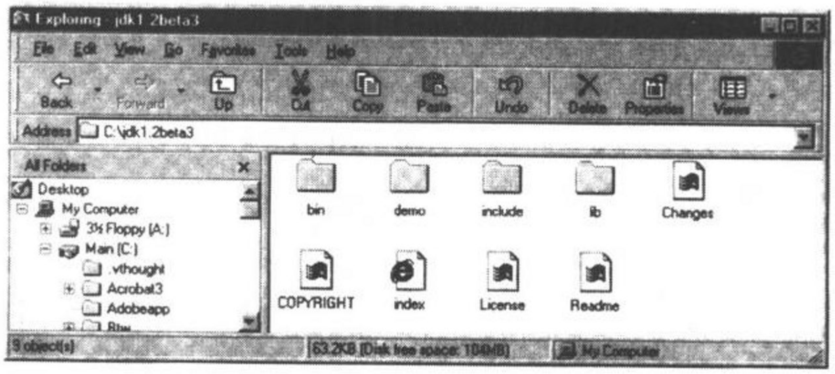


1.3 INSTALLING THE JDK

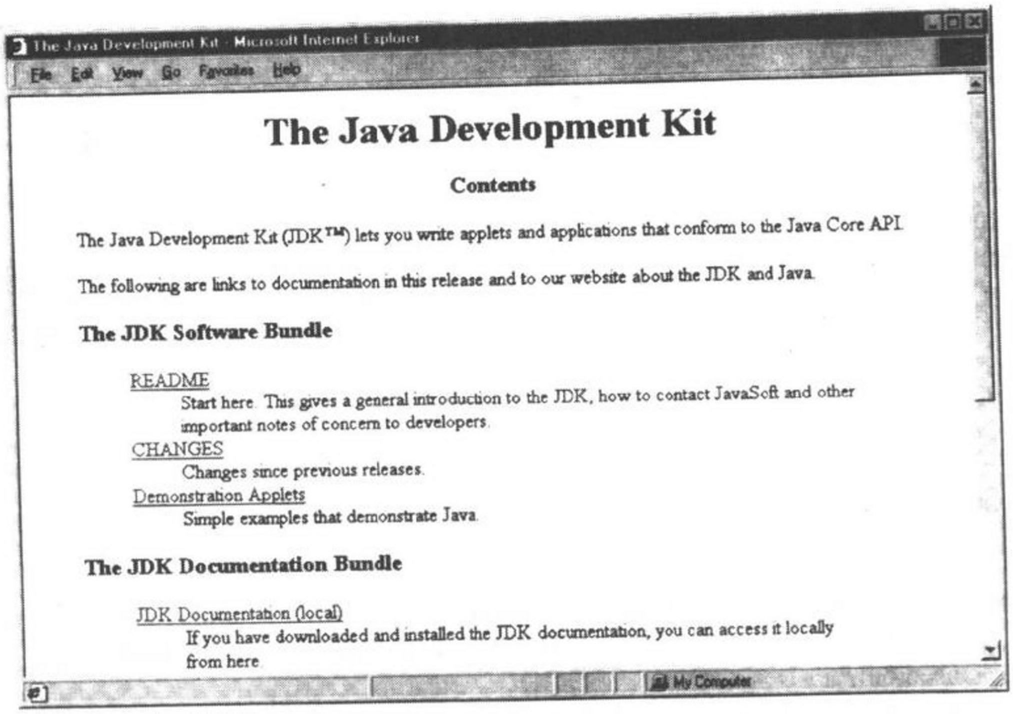
After you have downloaded the JDK you can install it simply by running the executable program that you downloaded. To do that, double-click on the icon that was placed on your Desktop.

jdk12-beta3-...

This only takes about a minute to install. Confirm all the suggested alternatives during the installation. During the installation process, the 15 megabytes gets decompressed into about 25 megabytes. The normal installation (in Microsoft Windows) would be into a new folder on your C: drive. When finished, it should look like this:



The `Readme` file is a text file that gives current information about the JDK and summarizes the contents of your `jdk` directory. The `index` file is a hypertext (web browser) file that outlines the JDK and provides browser-type access to its documentation, examples, and the JavaSoft web site. The `bin` directory contains the executable programs (binary files) that make up the JDK. The `lib` directory contains the library files for the Java language. The `include` directory contains the source code files that define the standard classes of the Java language. The `demo` directory contains over 20 demonstration programs.


The Java Development Kit - Microsoft Internet Explorer
File Edit View Go Favorites Help
My Computer

The Java Development Kit

Contents

The Java Development Kit (JDK™) lets you write applets and applications that conform to the Java Core API.

The following are links to documentation in this release and to our website about the JDK and Java.

The JDK Software Bundle

README
Start here. This gives a general introduction to the JDK, how to contact JavaSoft and other important notes of concern to developers.

CHANGES
Changes since previous releases.

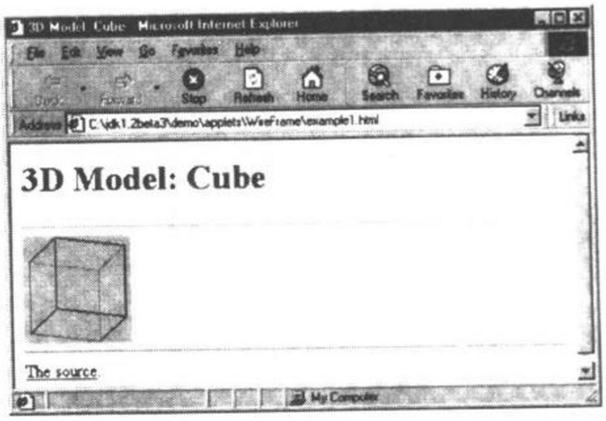
Demonstration Applets
Simple examples that demonstrate Java.

The JDK Documentation Bundle

JDK Documentation (local)
If you have downloaded and installed the JDK documentation, you can access it locally from here.

Double-click on the `index` file. This launches your web browser, displaying a web page entitled **The Java Development Kit**, as shown below.

To get an idea of what Java programs can do, click on the Demonstration Applets link. That opens a folder window that shows the contents of your `C:\jdk1.2beta3\demo\applets\` folder. Click on the folder labeled Wire Frame and then click on the file named example1. This launches a separate web page entitled "3D Model: Cube," which displays a wire frame cube. Use your mouse to drag a corner of the cube around within its picture frame and watch it rotate.


3D Model: Cube - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels
Address
Links
My Computer

C:\jdk1.2beta3\demo\applets\WireFrame\example1.html

3D Model: Cube

The source.

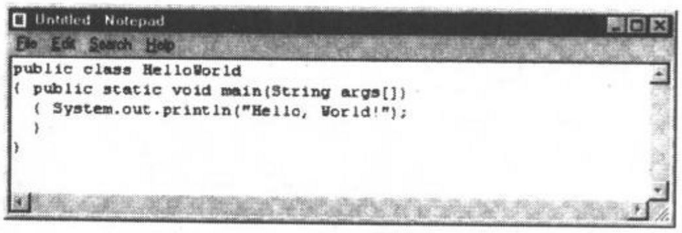
1.4 CREATING AND RUNNING A PROGRAM IN MICROSOFT WINDOWS

To create a Java program, you need to use an editor. You can use a word processor, such as Microsoft Word, but you have to be careful that the files you create are pure text files and that their file type is "`.java`". This section shows how to create a Java program using the simple Windows editor Notepad.

Start up the Notepad editor by selecting Programs > Accessories > Notepad from the Windows Start key. Then type the following four lines of Java code exactly as it is shown here:

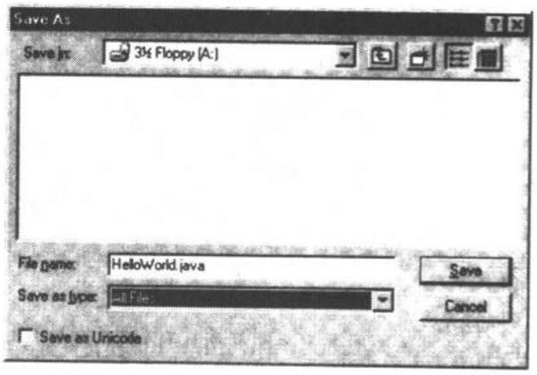
```
public class HelloWorld
{ public static void main(String[] args)
  { System.out.println("Hello, World!");
  }
}
```

Type capital letters as capital letters and lowercase letters as lowercase letters. (Java is *case-sensitive*.) Type the parentheses, brackets, and braces exactly as shown here, and don't miss the semicolon at the end of the third line. Your Notepad window should look like this:



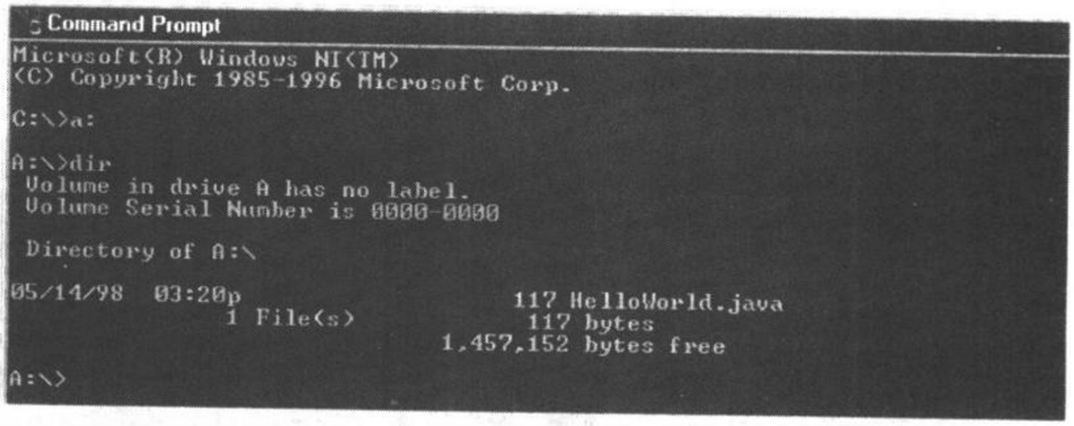
```
public class HelloWorld
{ public static void main(String args[])
  { System.out.println("Hello, World!");
  }
}
```

Save your file with the name `HelloWorld.java` as shown here.



Select All Files in the Save as type: field so that you can save the file with the correct file type. Notebook's default file type is `.txt`. But Java programs must have the file type `.java`.

Open a DOS command window by selecting Programs > Command Prompt from the Windows Start key. Use the DOS `cd` command to navigate to the folder that contains your Java program and then use the `dir` command to check that it is there and that it has the correct name.



```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>a:

A:\>dir
 Volume in drive A has no label.
 Volume Serial Number is 0000-0000

 Directory of A:\

05/14/98  03:20p                   117 HelloWorld.java
               1 File(s)            117 bytes
                              1,457,152 bytes free

A:\>
```

(In this demonstration, we are storing our Java programs on a floppy disk on the `A:\` drive.)

Execute the following DOS command to view the contents of your file:

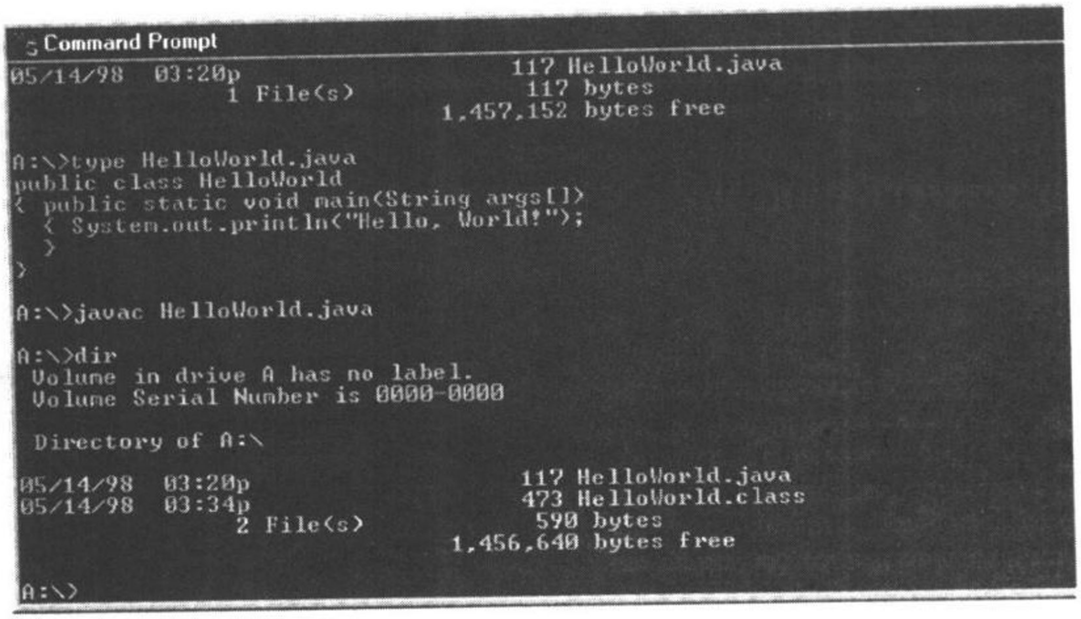
```
type HelloWorld.java
```

It should appear exactly as you typed it in the Notepad editor.

Now compile your program in the DOS window by executing the command

```
javac HelloWorld.java
```

If all goes well, the system should respond with another DOS prompt within a few seconds, thereby indicating that your program compiled successfully. If it did, then check your directory again to see that the compiler has produced a new file named `HelloWorld.class`.



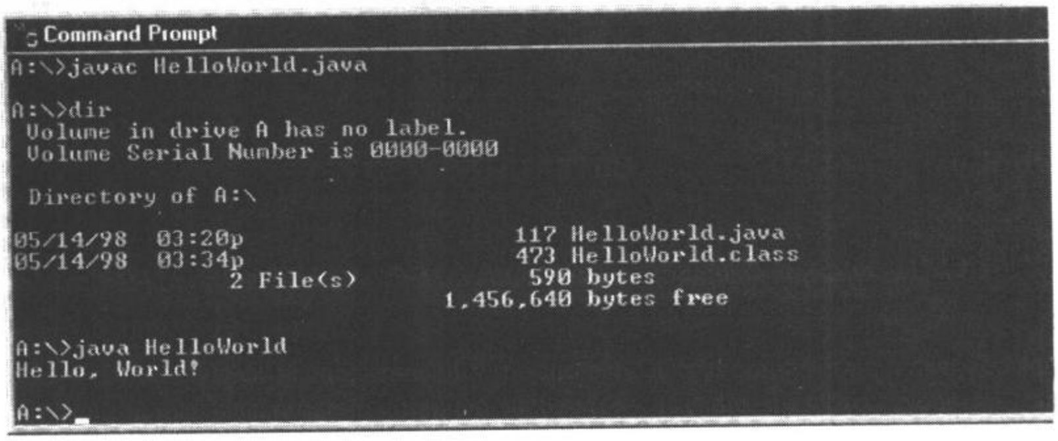
```
05/14/98  03:20p                     117 HelloWorld.java
                  1 File(s)            117 bytes
                                 1,457,152 bytes free

A:\>type HelloWorld.java
public class HelloWorld
{ public static void main(String args[])
  { System.out.println("Hello, World!");
  }
}

A:\>javac HelloWorld.java

A:\>dir
 Volume in drive A has no label.
 Volume Serial Number is 0000-0000

 Directory of A:\

05/14/98  03:20p                     117 HelloWorld.java
05/14/98  03:34p                     473 HelloWorld.class
                  2 File(s)            590 bytes
                                 1,456,640 bytes free

A:\>
```

This is the bytecode file that the JVM system uses to run your program. (Notice that it is more than four times the size of your source code file.)

Finally, execute the following command to run your program:

```
java HelloWorld
```

The system should respond by displaying the "Hello, World!" message:



```
A:\>javac HelloWorld.java

A:\>dir
 Volume in drive A has no label.
 Volume Serial Number is 0000-0000

 Directory of A:\

05/14/98  03:20p                     117 HelloWorld.java
05/14/98  03:34p                     473 HelloWorld.class
                  2 File(s)            590 bytes
                                 1,456,640 bytes free

A:\>java HelloWorld
Hello, World!

A:\>_
```

1.5 TROUBLESHOOTING

If you were able to get your program to run, skip to Section 1.6.

If your `HelloWorld.java` file did not show up in your folder when you executed the `dir` command, re-save it from the Notepad window.

If your `HelloWorld.java` file did show up in your folder but with the wrong name, re-save it from the Notepad window.

If your `HelloWorld.java` file did show up in your folder with the right name but its contents did not come out right when you executed the type HelloWorld.java command, go back to your Notepad window, correct the errors, and then re-save the file.

If the `javac HelloWorld.java` compile command did not work, try it again. Be sure you type "javac" (for "java compiler").

If the system does not know what the `javac` command is, then either the JDK is not installed correctly or the system does not know where it is installed. Use your Windows Explorer browser to find the folder where the JDK is installed. It should have a name like `jdk1.2` and be located on your `C:\` disk. Find the correct name for this folder, and then try executing the compile command with its path as a prefix, like this:

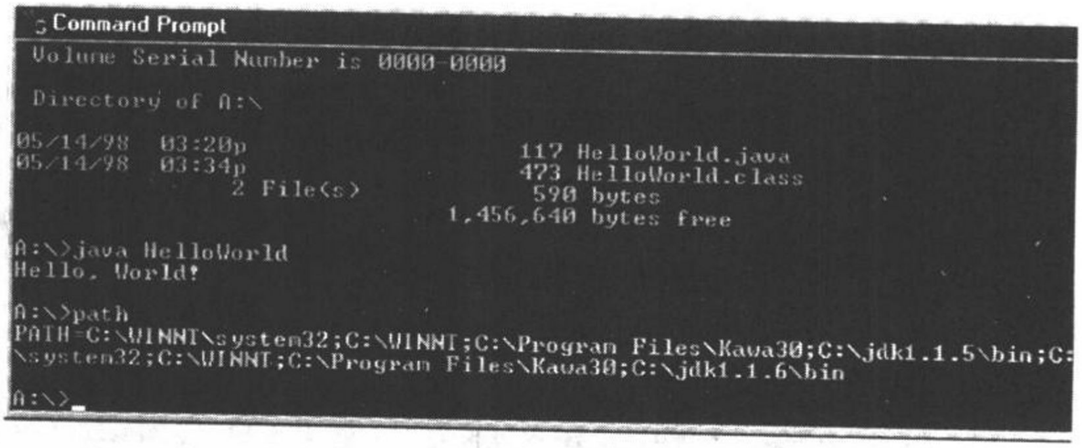
```
c:\jdk1.2\javac HelloWorld.java
```

If that works, then all you have to do is change the system `PATH` variable. If that does not work, then the JDK is probably not installed correctly. In that case, run the installation program again. If that doesn't work, start over again with a new download.

To determine whether the problem is with your system `PATH` variable, execute the following DOS command:

```
path
```

The system should respond like this:



```
Command Prompt
 Volume Serial Number is 0000-0000

 Directory of A:\

05/14/98  03:20p                       117 HelloWorld.java
05/14/98  03:34p                       473 HelloWorld.class
               2 File(s)                590 bytes
                                  1,456,640 bytes free

A:\>java HelloWorld
Hello, World!

A:\>path
PATH=C:\WINNT\system32;C:\WINNT;C:\Program Files\Kava30;C:\jdk1.1.5\bin;C:
\system32;C:\WINNT;C:\Program Files\Kava30;C:\jdk1.1.6\bin

A:\>_
```

The code that begins `PATH=C:\WIN...` is a listing of all the paths that the operating system checks to find the commands that you want to execute. The paths are separated by semicolons. One of them should include the name "`jdk`". If one does, then that is where the system is looking for the instructions on how to execute the `javac` command. If none of the paths includes the name "`jdk`", then you'll have to amend your system `PATH` variable.

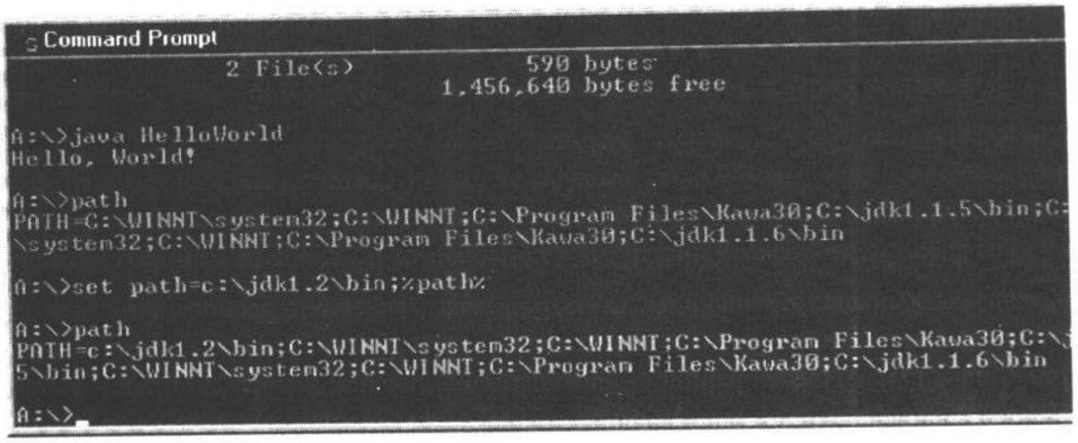
To amend your system `PATH` variable so that it includes the path to the JDK, execute the following DOS command:

```
set path=c:\jdk1.2\bin;%path%
```

and then execute the plain

```
path
```

command again to see if the PATH variable was amended correctly:



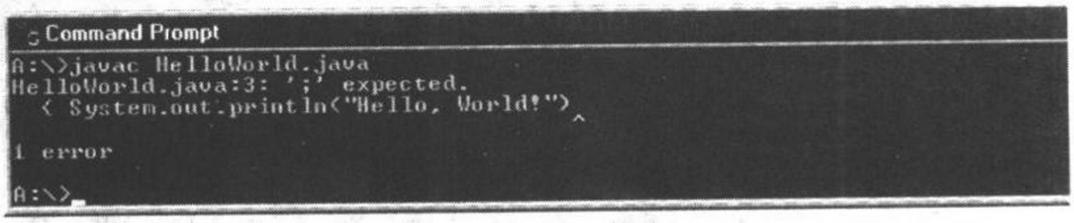
```
Command Prompt
                    2 File(s)              590 bytes
                                    1,456,640 bytes free

A:\>java HelloWorld
Hello, World!

A:\>path
PATH=C:\WINNT\system32;C:\WINNT;C:\Program Files\Kawa30;C:\jdk1.1.5\bin;C:
\system32;C:\WINNT;C:\Program Files\Kawa30;C:\jdk1.1.6\bin

A:\>set path=c:\jdk1.2\bin;%path%

A:\>path
PATH=c:\jdk1.2\bin;C:\WINNT\system32;C:\WINNT;C:\Program Files\Kawa30;C:\
5\bin;C:\WINNT\system32;C:\WINNT;C:\Program Files\Kawa30;C:\jdk1.1.6\bin

A:\>_
```

If it was, then you should now be able to execute the `javac` command successfully without using the prefix.

The instructions in the previous paragraph assume that the JDK is successfully installed in the folder named `jdk1.2`. If your JDK folder has a different name, then use that instead.

If it was necessary to amend your `PATH` variable, then you should add the same set path command to your `c:\autoexec.bat` file so that that system variable will be set correctly each time to restart your computer.

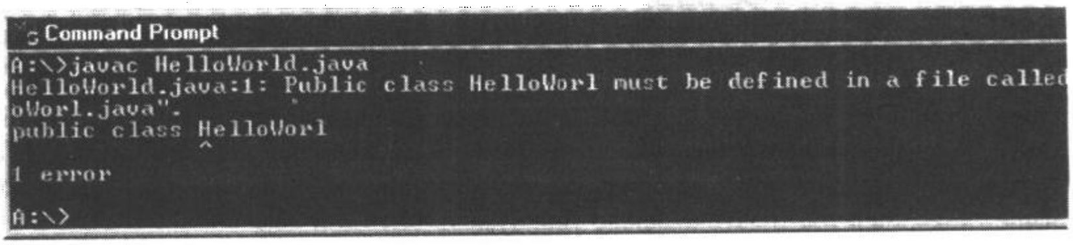
If the compiler displays error messages when you execute the `javac` command, then you have to go back into the editor and fix the errors. For example, if you omitted the semicolon at the end of the third line, then the compiler would respond like this:



```
Command Prompt
A:\>javac HelloWorld.java
HelloWorld.java:3: ';' expected.
  { System.out.println("Hello, World!")
                                        ^
1 error

A:\>_
```

The Java compiler is pretty good about locating and describing syntax errors like this.

If you see an error message like this



```
Command Prompt
A:\>javac HelloWorld.java
HelloWorld.java:1: Public class HelloWorl must be defined in a file called
oWorl.java".
public class HelloWorl
             ^
1 error

A:\>
```