COVERS J2SE VERSION 1.4

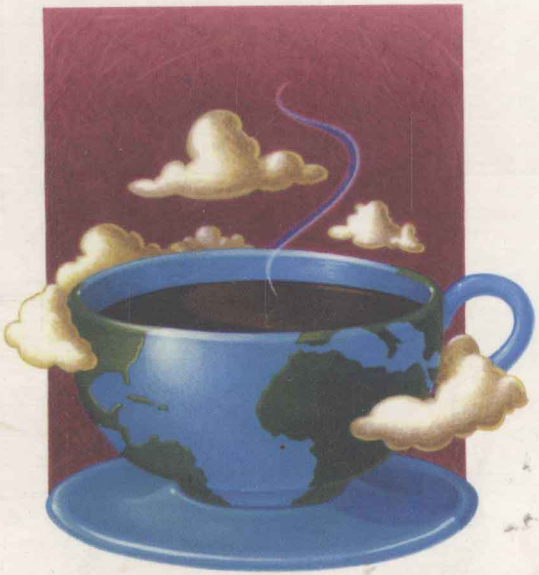
PEARSON Prentice Hall

core JAVA™ 2

Volume 1—Fundamentals, 6E

JAVA 2 核心编程

第1卷：基础篇 （第6版，影印版）



美国"软件开发"杂志第十三届图书效率大奖

Sun microsystems

（美） Cay S. Horstmann, Gary Cornell 著

清华大学出版社

Java 2 核心编程

第 1 卷：基础篇

（第 6 版，影印版）

（美） Cay S. Horstmann
Gary Cornell 著

清华大学出版社
北京

北京市版权局著作权合同登记号 图字: 01-2003-4881

本书封面贴有 Pearson Education(培生教育出版集团)激光防伪标签,无标签者不得销售。

图书在版编目(CIP)数据

Java 2 核心编程. 第 1 卷: 基础篇=Core Java 2, Volume 1—Fundamentals, 第 6 版/(美)霍斯特曼,(美)科奈尔著. —影印本.—北京: 清华大学出版社, 2003
ISBN 7-302-07198-5
Ⅰ.J… Ⅱ.①霍… ②科… Ⅲ.JAVA 语言—程序设计—英文 Ⅳ.TP312
中国版本图书馆 CIP 数据核字(2003)第 078277 号

出 版 者: 清华大学出版社 **地 址**: 北京清华大学学研大厦
http://www.tup.com.cn **邮 编**: 100084
社 总 机: 010-62770175 **客户服务**: 010-62776969

文稿编辑: 汤涌涛
封面设计: 立日新设计公司
印 刷 者: 世界知识印刷厂
装 订 者: 北京市密云县京文制本装订厂
发 行 者: 新华书店总店北京发行所
开 本: 148×210 **印张**: 23.5
版 次: 2003 年 9 月第 1 版 2004 年 2 月第 2 次印刷
书 号: ISBN 7-302-07198-5/TP·5240
印 数: 3001~5000
定 价: 49.00 元

The publisher offers discounts on this book when ordered in bulk quantities.
For more information, contact Corporate Sales Department, Prentice Hall PTR,
One Lake Street, Upper Saddle River, NJ 07458. Phone: 800-382-3419; FAX: 201-236-7141.
E-mail: corpsales@prenhall.com.

Editorial/production supervision: *Navta Associates*
Project coordinator: *Anne Trowbridge*
Cover design director: *Jerry Votta*
Cover designer: *Nina Scuderi*
Cover illustration: *Karen Strelecki*
Manufacturing manager: *Alexis R. Heydt*
Marketing manager: *Debby van Dijk*
Acquisitions editor: *Gregory G. Doench*

Sun Microsystems Press Publisher: *Michael Llwyd Alread*

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-047177-1

Sun Microsystems Press
A Prentice Hall Title

List of Tables, Code Examples, and Figures

Tables

1-1: The Growth of the Java Standard Edition API, 11
2-1: Java directory tree, 18
3-1: Java integer types, 39
3-2: Floating-point types, 40
3-3: Special characters, 41
3-4: Operator precedence, 49
3-5: Growth of an investment at different interest rates, 83
4-1: UML notation for class relationships, 93
7-1: Standard colors, 258
7-2: System colors, 260
8-1: Event handling summary, 298
8-2: Sample cursor shapes, 307
8-3: Predefined action table names, 317
8-4: Input map conditions, 319
9-1: The accessor methods of the `ButtonModel` interface, 340
9-2: MaskFormatter symbols, 361
9-3: Adding a Spring to an Over-constrained Component, 441
10-1: Applet positioning attributes, 503
10-2: `showDocument` arguments, 515
10-3: jar program options, 524
11-1: Timing data, 576
11-2: File handler configuration parameters, 584
11-3: Log file pattern variables, 584
11-4: HPROF options, 609
11-5: Debugging commands, 616
12-1: Basic character encodings (in `rt.jar`), 634
12-2: Extended Character Encodings (in `i18n.jar`), 634
12-3: Required character encodings, 697
12-4: Regular expression syntax, 700
12-5: Predefined character class names, 701

Code Examples

2-1: Welcome.java, 20
2-2: ImageViewer.java, 28
2-3: WelcomeApplet.html, 30
2-4: WelcomeApplet.java, 32
3-1: FirstSample.java, 38
3-2: InputTest.java, 57
3-3: Retirement.java, 65
3-4: Retirement2.java, 66
3-5: LotteryOdds.java, 70
3-6: BigIntegerTest.java, 75
3-7: LotteryDrawing.java, 80
3-8: CompoundInterest.java, 83
3-9: LotteryArray.java, 86
4-1: CalendarTest.java, 102
4-2: EmployeeTest.java, 106
4-3: StaticTest.java, 117
4-4: ParamTest.java, 122
4-5: ConstructorTest.java, 129
4-6: PackageTest.java, 134
4-7: Employee.java, 134
5-1: ManagerTest.java, 150
5-2: PersonTest.java, 160
5-3: EqualsTest.java, 167
5-4: ArrayListTest.java, 175
5-5: ReflectionTest.java, 185
5-6: ObjectAnalyzerTest.java, 190
5-7: ArrayGrowTest.java, 193
5-8: MethodPointerTest.java, 197
6-1: EmployeeSortTest.java, 203
6-2: TimerTest.java, 209
6-3: CloneTest.java, 213
6-4: InnerClassTest.java, 218
6-5: AnonymousInnerClassTest.java, 226
6-6: StaticInnerClassTest.java, 228
6-7: ProxyTest.java, 232
7-1: SimpleFrameTest.java, 239
7-2: CenteredFrameTest.java, 244
7-3: NotHelloWorld.java, 250
7-4: DrawTest.java, 256
7-5: FillTest.java, 261
7-6: FontTest.java, 266
7-7: ImageTest.java, 272
8-1: ButtonTest.java, 282
8-2: PlafTest.java, 290
8-3: Sketch.java, 302
8-4: MouseTest.java, 309
8-5: ActionTest.java, 320
8-6: MulticastTest.java, 325
8-7: CustomEventTest.java, 331
9-1: Calculator.java, 347
9-2: TextTest.java, 353
9-3: FormatTest.java, 363
9-4: TextAreaTest.java, 372
9-5: TextEditTest.java, 376
9-6: CheckBoxTest.java, 379
9-7: RadioButtonTest.java, 383
9-8: BorderTest.java, 386
9-9: ComboBoxTest.java, 390
9-10: SliderTest.java, 394
9-11: SpinnerTest.java, 400
9-12: MenuTest.java, 416
9-13: ToolBarTest.java, 422
9-14: BoxLayoutTest.java, 428
9-15: FontDialog.java, 434
9-16: SpringLayoutTest.java, 441
9-17: CircleLayoutTest.java, 447
9-18: OptionDialogTest.java, 455
9-19: DialogTest.java, 464
9-20: DataExchangeTest.java, 468
9-21: FileChooserTest.java, 478
9-22: ColorChooserTest.java, 486
10-1: NotHelloWorldApplet.java, 494
10-2: Calculator.html, 497
10-3: CalculatorApplet.java, 497
10-4: PopupCalculatorApplet.java, 501
10-5: Chart.java, 508
10-6: Bookmark.html, 516
10-7: Left.html, 517
10-8: Right.html, 517
10-9: Bookmark.java, 517
10-10: AppletFrame.java, 521
10-11: CalculatorAppletApplication.java, 522
10-12: ResourceTest.java, 529
10-13: WebStartCalculator.java, 537
10-14: CustomWorld.java, 547
10-15: SystemInfo.java, 550

10-16: PreferencesTest.java, 553
11-1: StackTraceTest.java, 570
11-2: ExceptTest.java, 572
11-3: ExceptionalTest.java, 578
11-4: LoggingImageViewer.java, 587
11-5: ConsoleWindow.java, 601
11-6: EventTracer.java, 603
11-7: EventTracerTest.java, 605
11-8: RobotTest.java, 606
11-9: WordCount.java, 612
11-10: BuggyButtonTest.java, 614
11-11: BuggyButtonFrame.java, 614
11-12: BuggyButtonPanel.java, 615
12-1: ZipTest.java, 642
12-2: DataFileTest.java, 651
12-3: RandomFileTest.java, 657
12-4: ObjectFileTest.java, 662
12-5: ObjectRefTest.java, 672
12-6: SerialCloneTest.java, 682
12-7: FindDirectories.java, 686
12-8: CRC.java, 692
12-9: NIOCRC.java, 692
12-10: RegexTest.java, 702
12-11: HrefMatch.java, 704

Figures

1-1: The Jmol applet, 9
2-1: Compiling and running `Welcome.java`, 19
2-2: Starting Sun ONE Studio, 21
2-3: The edit window of Sun ONE Studio, 22
2-4: The output window of Sun ONE Studio, 22
2-5: Error messages in Sun ONE Studio, 23
2-6: Starting a new program in Sun ONE Studio, 23
2-7: Compiling a program with Emacs, 24
2-8: Running a program from within Emacs, 25
2-9: Locating compilation errors in TextPad, 26
2-10: Running a Java program from TextPad, 26
2-11: Running the ImageViewer application, 27
2-12: The WelcomeApplet applet as viewed by the applet viewer, 30
2-13: Running the WelcomeApplet applet in a browser, 31
3-1: Legal conversions between numeric types, 48
3-2: The three panes of the API documentation, 54
3-3: Class description for the `String` class, 54
3-4: Method summary of the `String` class, 55
3-5: Detailed description of a `String` method, 55
3-6: An input dialog, 56
3-7: Flowchart for the `if` statement, 61
3-8: Flowchart for the `if/else` statement, 62
3-9: Flowchart for the `if/else if` (multiple branches), 63
3-10: Flowchart for the `while` statement, 64
3-11: Flowchart for the `do/while` statement, 67
3-12: Flowchart for the `for` statement, 68
3-13: Flowchart for the `switch` statement, 72
3-14: Copying an array variable, 78
3-15: Copying values between arrays, 78
3-16: A two-dimensional array, 85
4-1: A class diagram, 94
4-2: Procedural vs. OO programming, 95
4-3: Creating a new object, 97
4-4: Object variables that refer to the same object, 97
4-5: Returning a reference to a mutable data field, 112
4-6: Modifying a numeric parameter has no lasting effect, 119
4-7: Modifying an object parameter has a lasting effect, 120

4-8: Swapping object parameters has no lasting effect, 121
4-9: Changing the warning string in an applet window, 138
5-1: Employee inheritance hierarchy, 152
5-2: Inheritance diagram for `Person` and its subclasses, 158
6-1: Copying and cloning, 210
6-2: A shallow copy, 211
6-3: An inner class object has a reference to an outer class object, 217
7-1: The Windows look and feel of Swing, 237
7-2: The Motif look and feel of Swing, 237
7-3: The Metal look and feel of Swing, 238
7-4: The simplest visible frame, 239
7-5: Inheritance hierarchy for the `JFrame` and `JPanel` classes, 242
7-6: A simple graphical program, 246
7-7: The internal structure of a `JFrame`, 247
7-8: 2D rectangle classes, 253
7-9: The bounding rectangle of an ellipse, 254
7-10: Relationships between the shape classes, 254
7-11: Rectangles and ellipses, 255
7-12: Filled rectangles and ellipses, 261
7-13: Typesetting terms illustrated, 265
7-14: Drawing the baseline and string bounds, 266
7-15: Window with tiled graphics image, 271
8-1: Event notification, 279
8-2: A panel filled with buttons, 280
8-3: Switching the Look and Feel, 290
8-4: A window listener, 293
8-5: Inheritance diagram of the AWT event classes, 295
8-6: Relationship between event sources and listeners, 299
8-7: A sketch program, 302
8-8: A mouse test program, 305
8-9: Buttons display the icons from the Action objects, 318
8-10: All frames listen to the Close all command, 324
8-11: Using custom timer events to simulate rainfall, 330
9-1: Model and view of a text field, 336
9-2: Two separate views of the same model, 337
9-3: A window place, 338
9-4: Interactions between model, view, and controller objects, 339
9-5: A panel with three buttons, 341
9-6: A panel with six buttons managed by a flow layout, 342
9-7: Changing the panel size rearranges the buttons automatically, 342
9-8: Border layout, 343
9-9: A single button managed by a border layout, 345
9-10: A panel placed at the south end of the frame, 345
9-11: A calculator, 346
9-12: Text field example, 351
9-13: The `FormatTest` program, 363
9-14: A text area, 371
9-15: Testing text editing, 375
9-16: Check boxes, 379
9-17: A radio button group, 382
9-18: Testing border types, 386
9-19: A combo box, 389
9-20: Sliders, 393
9-21: Several variations of the `JSpinner` component, 398
9-22: A menu with a submenu, 406
9-23: Icons in menu items, 409
9-24: A checked menu item and menu items with radio buttons, 410
9-25: A pop-up menu, 411
9-26: Keyboard mnemonics, 413
9-27: Accelerators, 414
9-28: Disabled menu items, 415
9-29: A tool bar, 419
9-30: Dragging the tool bar, 420
9-31: Dragging the tool bar to another border, 420

9-32: Detaching the tool bar, 420
9-33: A tool tip, 421
9-34: Inheritance hierarchy for the `Component` class, 425
9-35: Box layouts, 428
9-36: Font dialog box, 431
9-37: Dialog box grid used in design, 431
9-38: A spring, 437
9-39: Summing springs, 437
9-40: Equally Spaced Buttons, 438
9-41: Springs and Struts, 439
9-42: Lining up Columns, 439
9-43: Horizontal springs attached to a component, 440
9-44: Circle layout, 446
9-45: Geometric traversal order, 451
9-46: An option dialog, 453
9-47: The OptionDialogTest program, 454
9-48: An About dialog box, 463
9-49: Password dialog box, 466
9-50: File chooser dialog box, 473
9-51: A file dialog with a preview accessory, 478
9-52: The "swatches" pane of a color chooser, 484
9-53: The HSB pane of a color chooser, 484
9-54: The RGB pane of a color chooser, 485
10-1: Applet inheritance hierarchy, 493
10-2: Viewing an applet in the applet viewer, 495
10-3: Viewing an applet in a browser, 496
10-4: A calculator applet, 497
10-5: A pop-up window inside a browser, 501
10-6: Applet alignment, 503
10-7: A chart applet, 507
10-8: A bookmark applet, 516
10-9: The calculator as an application, 519
10-10: The calculator as an applet, 519
10-11: Displaying a resource from a JAR file, 527
10-12: Launching Java Web Start, 533
10-13: The Calculator delivered by Java Web Start, 533
10-14: The Java Web Start Application Manager, 534
10-15: A Java Web Start Security Advisory, 534
10-16: The WebStartCalculator Application, 537
10-17: The customized Hello World program, 546
11-1: Exception hierarchy in Java, 559
11-2: A program that generates exceptions, 572
11-3: A log handler that displays records in a window, 584
11-4: The console window, 601
11-5: The `EventTracer` class at work, 602
11-6: Breakpoints in the Sun ONE Studio debugger, 619
11-7: Stopping at a Breakpoint, 620
12-1: Input and Output stream hierarchy, 625
12-2: Reader and Writer hierarchy, 626
12-3: A sequence of filtered stream, 628
12-4: The ZipTest program, 642
12-5: Two managers can share a mutual employee, 669
12-6: Here, Harry is saved three times, 670
12-7: An example of object serialization, 671
12-8: Objects saved in random order, 672
12-9: The graphical version of the `serialver` program, 680
12-10: Reading an object with fewer data fields, 681
12-11: Reading an object with more data fields, 682

Preface

To the Reader

In late 1995, the Java programming language burst onto the Internet scene and gained instant celebrity status. The promise of Java technology was that it would become the *universal glue* that connects users with information, whether that information comes from Web servers, databases, information providers, or any other imaginable source. Indeed, Java is in a unique position to fulfill this promise. It is an extremely solidly engineered language that has gained acceptance by all major vendors, except for Microsoft. Its built-in security and safety features are reassuring both to programmers and to the users of Java programs. Java even has built-in support that makes advanced programming tasks, such as network programming, database connectivity, and multithreading, straightforward.

Since 1995, Sun Microsystems has released five major revisions of the Java Software Development Kit. Over the course of the last 7 years, the Application Programming Interface (API) has grown from about 200 to just over 3,000 classes. The API now spans such diverse areas as user interface construction, database management, internationalization, security, and XML processing.

The book you have in your hand is the first volume of the sixth edition of the *Core Java* book. With the publishing of each edition, the book followed the release of the Java Software Development Kit as quickly as possible, and each time, we rewrote the book to take advantage of the newest Java features.

As with the previous editions of this book, we *still target serious programmers who want to put Java to work on real projects*. We still guarantee no nervous text or dancing tooth-shaped characters. We think of you, our reader, as a programmer with a solid background in a programming language. *But you do not need to know C++ or object-oriented programming*. Based on the responses we have received to the earlier editions of this book, we remain confident that experienced Visual Basic, C, or COBOL programmers will have no trouble with this book. (You don't even need any experience in building graphical user interfaces for Windows, Unix, or the Macintosh.)

What we do is assume you want to:

- Write real code to solve real problems;

and

- Don't like books filled with toy examples (such as kitchen appliances or fruit trees).

In this book you will find lots of sample code that demonstrates almost every language and library feature that we discuss. We kept the sample programs purposefully simple to focus on the major points, but, for the most part, they aren't fake and they don't cut corners. They should make good starting points for your own code.

We assume you are willing, even eager, to learn about all the advanced features that Java puts at your disposal. For example, we give you a detailed treatment of:

- Object-oriented programming
- Reflection and proxies
- Interfaces and inner classes
- The event listener model
- Graphical user interface design with the Swing UI toolkit
- Exception handling
- Stream input/output and object serialization

We *still* don't spend much time on the fun but less serious kind of Java programs whose sole purpose is to liven up your Web page. There are quite a few sources for this kind of material already—we recommend John Pew's book *Instant Java*, also published by Sun Microsystems Press.

Finally, with the explosive growth of the Java class library, a one-volume treatment of all the features of Java that serious programmers need to know is no longer possible. Hence, we decided to break the book up into two volumes. The first volume, which you hold in your hands, concentrates on the fundamental concepts of the Java language, along with the basics of user-interface programming. The second volume goes further into the enterprise features and advanced user-interface programming. It includes detailed discussions of:

- Multithreading
- Distributed objects
- Databases
- Advanced GUI components
- Native methods
- XML Processing
- Network programming
- Collection classes
- Advanced graphics
- Internationalization
- JavaBeans

When writing a book, errors and inaccuracies are inevitable. We'd very much like to know about them. But, of course, we'd prefer to learn about each of them only once. We have put up a list of frequently asked questions, bugs fixes, and workarounds in a Web page at `http://www.horstmann.com/corejava.html`. Strategically placed at the end of the FAQ (to encourage you to read through it first) is a form you can use to report bugs and suggest improvements. Please don't be disappointed if we don't answer every query or if we don't get back to you immediately. We do read all e-mail and appreciate your input to make future editions of this book clearer and more informative.

We hope that you find this book enjoyable and helpful in your Java programming.

About This Book

Chapter 1 gives an overview of the capabilities of Java that set it apart from other programming languages. We explain what the designers of the language set out to do and to what extent they succeeded. Then, we give a short history of how Java came into being and how it has evolved.

In Chapter 2, we tell you how to download and install the Java SDK from `http://java.sun.com/j2se`. We also describe how to download and install the Core Java program examples for

this book from `www.phptr.com/corejava`. Then we guide you through compiling and running three typical Java programs, a console application, a graphical application, and an applet.

Chapter 3 starts the discussion of the Java language. In this chapter, we cover the basics: variables, loops, and simple functions. If you are a C or C++ programmer, this is smooth sailing because the syntax for these language features is essentially the same as in C. If you come from a non-C background such as Visual Basic or COBOL, you will want to read this chapter carefully.

Object-oriented programming (OOP) is now in the mainstream of programming practice, and Java is completely object-oriented. Chapter 4 introduces encapsulation, the first of two fundamental building blocks of object orientation, and the Java language mechanism to implement it, that is, classes and methods. In addition to the rules of the Java language, we also give advice on sound OOP design. Finally, we cover the marvelous `javadoc` tool that formats your code comments as a set of hyperlinked web pages. If you are familiar with C++, then you can browse through this chapter quickly. Programmers coming from a non-object-oriented background should expect to spend some time mastering OOP concepts before going further with Java.

Classes and encapsulation are only one part of the OOP story, and Chapter 5 introduces the other, namely, *inheritance*. Inheritance lets you take an existing class and modify it according to your needs. This is a fundamental technique for programming in Java. The inheritance mechanism in Java is quite similar to that in C++. Once again, C++ programmers can focus on the differences between the languages.

Chapter 6 shows you how to use Java's notion of an *interface*. Interfaces let you go beyond the simple inheritance model of Chapter 5. Mastering interfaces allows you to have full access to the power of Java's completely object-oriented approach to programming. We also cover a useful technical feature of Java called *inner classes*. Inner classes help make your code cleaner and more concise.

In Chapter 7, we begin application programming in earnest. We show how you can make windows, how to paint on them, how to draw with geometric shapes, how to format text in multiple fonts, and how to display images.

Chapter 8 is a detailed discussion of the event model of the AWT, the *abstract windows toolkit*. (We discuss the event model that was added to Java 1.1, not the obsolete and simplistic 1.0 event model.) You'll see how to write the code that responds to events like mouse clicks or key presses. Along the way you'll see how to handle basic GUI elements like buttons and panels.

Chapter 9 discusses the Swing GUI toolkit in great detail. The Swing toolkit allows you to build a cross-platform graphical user interface. You'll learn all about the various kinds of buttons, text components, borders, sliders, list boxes, menus, and dialog boxes. However, some of the more advanced components are discussed in Volume 2.

After you finish Chapter 9, you finally have all mechanisms in place to write *applets*, those mini-programs that can live inside a Web page, and so applets are the topic of Chapter 10. We show you a number of useful and fun applets, but more importantly, we look at applets as a method of *program deployment*. We then describe how to package applications in JAR files, and how to deliver applications over the internet with the Java Web Start mechanism. Finally, we explain how Java programs can store and retrieve configuration information once they have been deployed.

Chapter 11 discusses *exception handling*, Java's robust mechanism to deal with the fact that bad things can happen to good programs. For example, a network connection can become unavailable in the middle of a file download, a disk can fill up, and so on. Exceptions give you an efficient way of separating the normal processing code from the error handling. Of course, even after hardening your program by handling all exceptional conditions, it still might fail to work as expected. In the second half of this chapter, we give you a large number of useful debugging tips. Finally, we guide you through sample sessions with various

tools: the JDB debugger, the debugger of an integrated development environment, a profiler, a code coverage testing tool, and the AWT robot.

We finish the book with input and output handling. In Java, all I/O is handled through so-called *streams*. Streams let you deal in a uniform manner with communicating with any source of data, such as files, network connections, or memory blocks. We include detailed coverage of the reader and writer classes, which make it easy to deal with Unicode. We show you what goes on under the hood when you use the object serialization mechanism, which makes saving and loading objects easy and convenient. Finally, we cover several libraries that have been added to SDK 1.4: the "new I/O" classes that contain support for advanced and more efficient file operations, and the regular expression library.

An appendix lists the Java language keywords.

Conventions

As is common in many computer books, we use `courier type` to represent computer code.

There are many C++ notes that explain the difference between Java and C++. You can skip over them if you don't have a background in C++ or if you consider your experience with that language a bad dream of which you'd rather not be reminded.

Notes and tips are tagged with "note" and "tip" icons that look like these.

When there is danger ahead, we warn you with a "Caution" icon.

Java comes with a large programming library or Application Programming Interface (API). When using an API call for the first time, we add a short summary description tagged with an API icon at the end of the section. These descriptions are a bit more informal, but we hope also a little more informative than those in the official on-line API documentation. We now tag each API note with the version number in which the feature was introduced, to help the readers who don't use the "bleeding edge" version of Java.

Programs whose source code is on the web are listed as examples, for instance **Example 2–4: WelcomeApplet.java.**

Sample Code

The web site for this book at `http://www.phptr.com/corejava` contains all sample code from the book, in compressed form. You can expand the file either with one of the familiar unzipping programs or simply with the `jar` utility that is part of the Java Software Development Kit. See Chapter 2 for more information about installing the Software Development Kit and the sample code.

Acknowledgments

Writing a book is always a monumental effort, and rewriting doesn't seem to be much easier, especially with continuous change in Java technology. Making a book a reality takes many dedicated people, and it is my great pleasure to acknowledge the contributions of the entire Core Java team.

A large number of individuals at Prentice-Hall PTR, Sun Microsystems Press, and Navta Inc. provided valuable assistance, but they managed to stay behind the scenes. I'd like them all to know how much I appreciate their efforts. As always, my warm thanks go to my editor, Greg Doench of Prentice-Hall PTR, for steering the book through the writing and production process, and for allowing me to be blissfully unaware of the existence of all those folks behind the scenes. My thanks also to my co-author of earlier editions, Gary Cornell, who has since moved on to other ventures.

Thanks to the many readers of earlier editions who reported embarrassing errors and made lots of thoughtful suggestions for improvement. I am particularly grateful to the excellent reviewing team that went over the manuscript with an amazing eye for detail and saved me from many more embarrassing errors.

Reviewers this and earlier editions include Chuck Allison (Contributing Editor, *C/C++ Users Journal*), Alec Beaton (PointBase, Inc.), Joshua Bloch (Sun Microsystems), David Brown, Dr. Nicholas J. De Lillo (Manhattan College), Rakesh Dhoopar (Oracle), David Geary, Angela Gordon (Sun Microsystems), Dan Gordon (Sun Microsystems), Rob Gordon, Cameron Gregory (olabs.com), Marty Hall (The Johns Hopkins University Applied Physics Lab), Vincent Hardy (Sun Microsystems), Vladimir Ivanovic (PointBase, Inc.), Jerry Jackson (ChannelPoint Software), Tim Kimmet (Preview Systems), Chris Laffra, Charlie Lai (Sun Microsystems), Doug Langston, Doug Lea (SUNY Oswego), Gregory Longshore, Bob Lynch, Mark Morrissey (The Oregon Graduate Institute), Mahesh Neelakanta (Florida Atlantic University), Paul Philion, Blake Ragsdell, Stuart Reges (University of Arizona), Peter Sander (ESSI University, Nice, France), Paul Sevinc (Teamup AG), Devang Shah (Sun Microsystems), Bradley A. Smith, Christopher Taylor, Luke Taylor (Valtech), George Thiruvathukal, Kim Topley, Janet Traub, Peter van der Linden (Sun Microsystems), and Burt Walsh.

Cay Horstmann

San Francisco, 2002

Contents

Preface, xvii
To the Reader, xvii
About This Book, xviii
Conventions, xx
Sample Code, xx

Acknowledgments, xxi

Chapter 1

An Introduction to Java, 1
Java as a Programming Tool, 2
Advantages of Java, 2
The Java "White Paper" Buzzwords, 3
Simple, 4
Object Oriented, 4
Distributed, 5
Robust, 5
Secure, 5
Architecture Neutral, 6
Portable, 7
Interpreted, 7
High Performance, 7
Multithreaded, 8
Dynamic, 8
Java and the Internet, 8
A Short History of Java, 9
Common Misconceptions About Java, 11

Chapter 2

The Java Programming Environment, 15
Installing the Java Software Development Kit, 15

Setting the Execution Path, 16
Installing the Library Source and Documentation, 17
Installing the Core Java Program Examples, 17
Navigating the Java Directories, 17
Development Environments, 18
Using the Command Line Tools, 19
Troubleshooting Hints, 20
Using an Integrated Development Environment, 21
Locating Compilation Errors, 22
Compiling and Running Programs from a Text Editor, 24
Graphical Applications, 27
Applets, 29

Chapter 3

Fundamental Programming Structures in Java, 35
A Simple Java Program, 35
Comments, 38
Data Types, 39
Integers, 39
Floating-Point Types, 40
The Character Type, 41
The `boolean` *Type, 42*
Variables, 42
Assignments and Initializations, 43
Constants, 43
Operators, 44
Increment and Decrement Operators, 45
Relational and `boolean` *Operators, 45*
Bitwise Operators, 46
Mathematical Functions and Constants, 47
Conversions Between Numeric Types, 47
Casts, 48
Parentheses and Operator Hierarchy, 49
Strings, 49
Concatenation, 50
Substrings, 50
String Editing, 50
Testing Strings for Equality, 52
Reading the On-line API Documentation, 53
Reading Input, 56
Formatting Output, 57
Control Flow, 60
Block Scope, 60
Conditional Statements, 61
Indeterminate Loops, 64
Determinate Loops, 68
Multiple Selections—the `switch` *Statement, 70*
Breaking Control Flow, 71
Big Numbers, 74
Arrays, 76

Array Initializers and Anonymous Arrays, 77
Copying Arrays, 77
Command Line Parameters, 79
Sorting an Array, 79
Multidimensional Arrays, 82
Ragged Arrays, 85

Chapter 4

Objects and Classes, 89
Introduction to Object-Oriented Programming, 89
The Vocabulary of OOP, 91
Objects, 91
Relationships Between Classes, 92
Contrasting OOP with Traditional Procedural Programming Techniques, 94
Using Existing Classes, 96
Objects and Object Variables, 96
The `GregorianCalendar` Class of the Java Library, 98
Building Your Own Classes, 104
An `Employee` Class, 104
Using Multiple Source Files, 107
Analyzing the `Employee` Class, 108
First Steps with Constructors, 108
The Methods of the `Employee` Class, 110
Method Access to Private Data, 113
Private Methods, 113
Final Instance Fields, 113
Static Fields and Methods, 114
Static Fields, 114
Constants, 114
Static Methods, 115
Factory Methods, 116
The `main` Method, 116
Method Parameters, 118
Object Construction, 124
Overloading, 124
Default Field Initialization, 124
Default Constructors, 125
Explicit Field Initialization, 125
Parameter Names, 126
Calling Another Constructor, 127
Initialization Blocks, 127
Object Destruction and the `finalize` Method, 131
Packages, 131
Using Packages, 132
Documentation Comments, 139
How to Insert Comments, 139
Class Comments, 139
Method Comments, 140
Field Comments, 140
General Comments, 141