Preface

This book can be used for a one-term or two-term course in Java and for self-study. Chapters 9 and 10 have sufficient material for an introductory course in network programming and distributed systems. The book assumes no prior knowledge of Java or object-oriented programming, but the book does assume programming experience. The reader should be familiar with general programming concepts such as *variable*, *data type*, and *statement*; programming constructs such as statement sequencing, function calls, tests, and loops; and programming practices such as design, coding, and testing. The book makes extensive use of examples, tables, figures, self-study exercises, sample applications, lists of common programming errors and safe practices, and programming exercises. Throughout the book, my goal is to write clear code and to illustrate sound programming practices. All of the code in the book has been compiled and run under Java 2. The book also draws many of its examples from Java's own standard classes so that the Java packages are introduced in a natural, gradual way.

The Special Challenges of Java

This book is based on courses that I have been teaching regularly and on my experience in writing textbooks, with Richard Johnsonbaugh, on C and C++. The Java courses serve three different audiences, which has been helpful in developing pedagogical techniques for presenting the language. One audience consists of undergraduate majors in computer science and information systems with prior coursework in another language, typically C++ or Visual Basic. The second audience consists of graduate students with strong programming skills and fluency in C/C++ or ML. The third audience consists of professional programmers with experience in various languages such as assembler, C, Perl, and COBOL. Despite differences in background, programmers intent on learning Java face challenges that reflect the language's most attractive features:

- Java is a *general-purpose language* with the rich assortment of data types, operators, control structures, and standard packages that befits this status. Although the book offers full coverage of Java, it does so without overwhelming the reader. I focus first on the relatively simple cases before moving to the more complicated ones. To take a basic example, Java has three loop constructs: the `for` loop, the `while` loop, and the `do while` loop. Of the three, the `while` loop has the simplest syntax and semantics. The `while` loop is thus presented first and the other loop structures are then clarified with reference to it. To take a more advanced example, Java supports graphics through both the Abstract Window Toolkit (AWT) and the Swing set. Although the Swing set is more flexible and powerful, the Abstract Window Toolkit is simpler; hence, the graphics coverage starts with basic AWT examples then moves to Swing set examples.
- Java is an *object-oriented programming language*. Although Java has primitive types such as `int` and `double`, every program requires at least one class. All functions are encapsulated in classes as either constructors or methods, and variables are either encapsulated fields or local variables in constructors and methods. Java has only single inheritance for classes but a class may implement arbitrarily many interfaces; and Java supports multiple inheritance for interfaces. Java also has abstract as well as concrete classes. Further, methods and fields can be associated either with a class itself (`static` methods and fields) or with class instances or objects (non`static` methods and fields). Object-oriented constructs such as polymorphism are widely used within Java's own standard classes. Although Java is an elegant object-oriented language, special attention still must be given to the reader who comes to Java from an exclusively or even primarily procedural background. This book addresses object orientation in two ways:
  - The first chapter clarifies the basic concepts, constructs, and benefits of object-oriented design and programming. The chapter explains key terms such as *class*, *object*, *information hiding*, *encapsulation*, *inheritance*, and *message passing*. The chapter also has a section on UML to introduce modern approaches to object-oriented design.
  - The book's short programs and longer sample applications present object-oriented programming as a natural and intuitive approach to programming. The goal is to illustrate the *benefits to the programmer* that object orientation brings.
- Java is a *modern programming language* with standard packages to support graphics, networking, security, persistence, database, reflection, components—and more. Java is large and growing. The standard `java` packages have been augmented by the `javax` packages (the `x` stands

for extension) and by many excellent commercial packages. The first seven chapters focus on core language features and the last three chapters then illustrate how these core features support networking in many forms (e.g., sockets, applets, servlets, remote method invocation, and object request brokers), database, security, and component-based programming. All ten chapters draw frequent examples from Java's own standard classes to give the reader an on-going introduction to Java's rich class libraries.

Chapter Overviews

The book has ten chapters, all of which include self-study exercises at the end of sections. Solutions to odd-numbered section exercises are included in the book, and the instructor CD-ROM has solutions to the even-numbered section exercises. All of the chapters except for the first have programming exercises and a list of common programming errors and safe practices. The book has

- Over 600 end-of-section exercises with answers to odd-numbered exercises.
- Over 140 programming exercises.
- 13 major sample applications and over 150 additional full programs.

Many chapters have a Java Postscript to handle special topics such as the 2's complement representation of integers, the `volatile` modifier, and deprecated `Thread` methods. The chapters include short programs as well as longer sample applications. All of the source code for programs and sample applications is available on the book's CD-ROM and at the Web site. The chapters can described as follows:

- **Chapter 1: Object-Oriented Programming**. This chapter explains the basic concepts and advantages of object-oriented design and programming, which are contrasted with top-down design and procedural programming. Central concepts such as *class*, *object*, *inheritance*, *polymorphism*, *abstract data type*, *message passing*, *interface*, and *component* are clarified and illustrated. Although the chapter includes some brief Java examples, the emphasis is on object-oriented programming in general. The chapter includes a section on UML with an overview of the modeling language and various examples of UML diagrams. The section also indicates how UML models can guide coding.
- **Chapter 2: Introductory Programs**. This chapter provides a series of short but realistic programs so that the reader is exposed at once to the "look and feel" of Java code. The sections and the section exercises suggest ways in which the programs can be adapted and expanded so that the reader can gain quick experience with Java programming. The chapter begins with a traditional *Hello, world!* program and then

introduces other short programs to illustrate variables of primitive and class types, variable declarations and assignments, control structures such as tests and loops, constructor and method invocation, and arrays. Writing, documenting, compiling, and running programs is explained carefully. The chapter assumes that the reader compiles and executes from the command line using the JDK for Java 2.

Some of the programs are paired with one program in the pair processing randomly generated numbers and the other reading data from a disk file. The chapter's second introductory program called `BigAndSmall` is an example. The program selects the largest and smallest integers from a sequence. In one version, the integers are randomly generated using the standard `java.util.Random` class; in a second version, the integers are read from a local disk file. The two versions together provide the reader with early, alternative examples of generating input data. A similar approach is taken with output: one program in a pair writes to the standard output, whereas the other program writes to a local disk file.

The chapter has a section dedicated to strings, introducing both the `String` and the `StringBuffer` classes. The section on the `String` class also emphasizes how the writing of a *test client* facilitates learning the language.

The three remaining sections cover programmer-defined classes, further basics of the `java.io` package, and the three sample utility classes: `Vector`, `Hashtable`, and `StringTokenizer`. The sample programs again are short but perform familiar, realistic programming tasks such as sorting.

- **Chapter 3: Programs and Packages**. This chapter explains the role of the Java Virtual Machine and the relationships among program types such as *application*, *applet*, *servlet*, and *bean*. The chapter reviews and extends the coverage of source (*java*) and compiled (*class*) files. The chapter also introduces the `package` statement and clarifies the `CLASSPATH` environment variable. The use of subdirectories as subpackages is covered as well. The chapter has a sample application with a programmer-defined package. Many reference figures, including lists of the standard `java` and `javax` packages, are included for convenience.
- **Chapter 4: Language Fundamentals**. This chapter covers language fundamentals, including identifiers, primitive and class data types, control structures, operators, arrays, and exceptions. The chapter has a section to review constructors and methods and then to extend coverage of these key topics. The chapter is organized carefully into sections and subsections so that the chapter can be used as a reference as well as an introduction to basic Java. My assumption is that the reader would return repeatedly to selected sections in the chapter. For instance, the

reader might delay a careful reading the section on the `switch` construct until a programming need arises. Some specialized language features such as the bitwise operators are explained in the Java Postscript rather than the chapter's main body. The chapter includes a sample application.

- **Chapter 5: Classes**. This chapter provides a comprehensive and technical coverage of classes. The chapter examines class and member scope, information hiding, encapsulation, constructors, methods, and fields. The chapter explains how technical aspects of class semantics can be used to achieve practical goals. For example, a subsection explains how selective constructor definition can be used to restrict object construction. The chapter also examines the key role of the no-argument constructor.

  The chapter includes a section on using class libraries such as Java's own standard libraries. This section reviews the use of test clients to gain fluency in a class, and underscores again the relationship between the *exposed* and *hidden* in a class.

  The chapter's first sample application provides two classes, `BasicInput` and `BasicOutput`, that illustrate the usefulness of *wrapper classes*. The two classes support the high-level input and output with intuitive constructors and methods. For instance, `BasicInput` objects can be constructed with either the standard input or a disk file as the source. Methods such as `getRecord` and `getDouble` are straightforward. This sample application, together with the introductory programs in Chapter 2 that use the `java.io` package for both binary and character input/output, provide sufficient examples to write programs that perform realistic input and output operations.

  The second sample application has a graphical component to illustrate how graphics is integrated into the language. The early use of graphics underscores that a program can be furnished with graphical components even before the full details of Java's graphics packages are mastered.

- **Chapter 6: Inheritance, Interfaces, and Abstract Classes**. This chapter extends the material covered in Chapter 5 by explaining inheritance, interfaces, and abstract classes. The chapter introduces the technical aspects of polymorphism and underscores its power with a series of short programs and a sample application on polymorphic input and output. The chapter goes into the details of constructors under inheritance, method overriding, and the use of interfaces in object-oriented programming.

  The chapter explains the differences between interfaces and classes, underscoring the object-oriented dictum about "programming to the interface." Examples of standard Java interfaces and programmer-defined interfaces are included. The chapter's last section introduces abstract

classes by contrasting these with concrete classes and interfaces. The section emphasizes that abstract classes are bona fide classes with special uses.

- **Chapter 7: Graphics and Event Handling**. This chapter begins with an overview of the Abstract Window Toolkit (AWT) and the Swing set, event-driven programming, and the Java event model. The chapter emphasizes common features between the AWT and the Swing set by focusing on fundamental constructs such as *container* and *component*. The chapter discusses the relationship between "heavyweight" and "lightweight" components and how this distinction relates to the AWT and the Swing set. A section on the model-view-controller architecture presents a foundational view of Swing set graphics. The chapter's many programs and two sample applications illustrate framed windows, dialogs, fonts, colors, layout managers, and controls such as buttons, menus, menu bars and tool bars, lists, checkboxes, and scrollbars. Basic drawing and images are also covered. The emphasis throughout is on practical approaches to graphics programming. For this reason, many short programs are used to focus on particular topics such as closing windows or using popup menus.

  The chapter emphasizes Swing set graphics wherever feasible but uses the relatively simpler AWT to illustrate some key ideas. The two sample applications introduce the high-level Swing set components `JTree` and `JTable`.

- **Chapter 8: Three Interfaces**. This chapter examines three key interfaces: `Cloneable`, `Serializable`, and `Runnable`. The first section explains the differences among object construction, the copying of object references, and cloning. The section discusses and illustrates the dangers of cloning objects whose fields include object or array references. There are subsections on overriding the default `clone` method, disabling cloning, and cloning arrays.

  The chapter's second section introduces serialization and object persistence. The section covers not only the basics but also technical details such as serialization for objects whose superclass does not implement `Serializable`, the serial version number, and the dangers of serializing the same object repeatedly to the same stream. The difference between serialization and writing primitive types to binary streams is clarified. Customized serialization is examined and motivated through a series of related examples, all of which are short but complete programs. There are subsections on disabling serialization and implementing `Externalizable`. A sample application on a serializable time card is included to review and consolidate the material. Chapter 9 on networking extends the discussion by covering serialization and sockets, and Chapter 10 covers the relationship between serialization and beans.

The third section offers a comprehensive introduction to multithreading. The section begins with a detailed examination of the differences between single-threaded and multithreaded applications, using full program examples to illustrate the benefits and basics of multithreading. The section covers thread priorities, the distinction between user and daemon threads, the relationship between `start` and `run`, the recommended way to stop threads, and the use of the `join` method. The section also explains thread groups. Once the basics have been examined, the section then illustrates and discusses the need for thread synchronization. Issues of deadlock, starvation, and fairness are covered in this section and in the sample application on the dining philosophers problem. Basic concepts such as *critical section*, *mutual exclusion*, and *lock* are explained and then illustrated with Java constructs. Chapter 9 on networking extends the discussion with examples of multithreaded servers.

- **Chapter 9: Network Programming**. This chapter covers networking in Java. The chapter begins with an overview section on networking basics, in particular on the TCP/IP protocol suite. Addresses, packets, transport protocols, sockets, firewalls, proxy servers, and other fundamental concepts are explained. A section on sockets follows. This section covers client `Sockets`, `ServerSockets`, and `DatagramSockets` with various examples. `MulticastSockets` are also clarified. For motivation, the chapter's examples are full programs, some of which implement familiar utilities such as a port tester and a *finger* program. The section highlights the power of serialization over sockets. A sample application illustrates a multithreaded server.

  The chapter's third section presents a thorough coverage of applets, including the issue of *sandbox security*. The section begins with elementary examples that can be adapted readily for experimentation. There is a multimedia applet and a discussion of applets and *jar* files. The section illustrates how applets can communicate with one another and how programs other than Web browsers can serve as host programs for applets. There is a sample program, a Java *application*, that downloads and then displays an applet in the application's own framed window. This section also highlights Java's support for URLs and other networking constructs. Although applets can be run on a standalone machine, this section emphasizes their usefulness in a distributed, client/server system. A sample application shows how an applet order form can use a socket to send information back to the server.

  The fourth section covers RMI. After motivating RMI, the section provides a step-by-step explanation of setting up an RMI server and client. The section explains the role of the registry and RMI activation. A final subsection introduces Jini, clarifying its relationship to RMI. A sam-

ple application on matrix algebra operations reviews and extends the RMI material. In particular, the sample application offers a realistic example of how RMI could be used in a distributed system.

The chapter's final section covers object-request brokers in general and CORBA in particular. Like the RMI section, this section offers a step-by-step explanation of setting up a CORBA server and client. The section covers the IDL file, the *idltojava* utility, CORBA modules and interfaces, language and location transparency, naming services, the dynamic invocation interface, and IIOP with respect to the convergence of RMI and CORBA technologies. The Java Postscript clarifies how an applet can be a CORBA client.

The chapter explains how distributed applications can be developed and tested on a standalone machine using the *localhost* IP address. This chapter together with Chapter 10 has sufficient material for an introductory course in network programming or distributed systems.

- **Chapter 10: Selected Topics**. This chapter covers special topics divided into four sections. The first section is devoted to component-based programming using Java *beans*. The section underscores how bean technology, including Enterprise Java Beans, leverages basic Java constructs such as serialization, interfaces, properties as pairs of *get*/*set* methods, and the event model. The section introduces and clarifies the *beanbox* utility for developing and testing beans. The section explains how property change events can be used for bean interaction.

  The second section deals with security and cryptography. The security roles of the the compiler, the bytecode verifier, and the security manager are examined. The section emphasizes the use of high-level security constructs such as the access controller and policy files. The subsection on security offers several illustrations of *permissions* for implementing security. Some examples such as the ones on sandbox security review and extend earlier sections. The subsection on cryptography first presents an overview of Java library support and then focuses on authentication. The relationships among message digests, public and private keys, and digital signatures is explained. The approach is a step-by-step discussion and illustration of how digital signatures are used on the sender and the receiver sides. Several short examples and one longer one illustrate Java support for authentication.

  The third section, on reflection, is also example-based. This section first presents an overview of Java's support for run-time class information and then illustrates with several examples. For instance, the section shows how basic information about a class can be reconstructed from a *class* file using reflection technology. The section also extends the discussion of beans by showing how reflection technology underlies the dynamic construction of property sheets in utilities such as the *beanbox*.

The fourth section covers servlets and database. Together with the sample application on database webification, this material extends the network programming covered in Chapter 9. After introducing servlet basics, the fourth section offers several short examples that the reader can extend. A subsection then explains how JDBC works and how database and servlet technology are commonly integrated. The examples use the sample Northwind database that comes with Microsoft's Access relational database management system. However, the examples are sufficiently modular so that they could be adapted straightforwardly to other databases. The coverage of JDBC emphasizes the core features such as *connection*, *query*, and *result set*. The sample application then illustrates the integration of servlet and database technology. In the sample application, one servlet presents a list of products in HTML. After the user selects a product, a second servlet generates an HTML list of customers who purchased the product. The data for both lists resides in a database. The application again is designed to be readily adaptable.

Chapter Structure

The basic chapter organization is as follows:

Contents
Overview
Section
Section Exercises
Section
Section Exercises
...
Java Postscript
Common Errors and Safe Practices
Programming Exercises

Chapters 3 through 10 have 13 sample applications. A sample application section contains a statement of a problem, sample input and output, a solution to the problem, and a Java implementation of a solution to the problem. The section concludes with an extended discussion of the Java implementation and program development. The sample applications include the following:

- Random number generation through a wrapper class (Section 3.4)
- Basic input and output classes (Section 5.4)
- A graphical utility for file copying (Section 5.5)
- Polymorphic input and output operations (Section 6.3)
- Graphical directory assistance (Section 7.4)

- A graphical table editor (Section 7.6)
- A serializable time card (Section 8.3)
- A multithreaded simulation of the dining philosophers problem (Section 8.5)
- A socketed applet with a membership form (Section 9.5)
- Matrix algebra operations using RMI (Section 9.7)
- Database webification using servlets (Section 10.5)

The *Java Postscript* sections discuss highly specialized parts of the language and give additional technical details about language features.

The *Common Errors and Safe Practices* sections highlight those aspects of the language that are easily misunderstood.

The book contains more than 140 programming exercises drawn from a wide variety of applications. The programming exercises differ in difficulty from the relatively straightforward to the highly challenging.

About This Book

This book includes:

- Examples and exercises that cover a wide range of applications.
- Motivating real-world applications.
- A broad variety of programming exercises. The book contains over 140 programming exercises.
- End-of-chapter lists of common programming errors and safe practices.
- Exercises at the ends of sections so that readers can check their mastery of the sections. The book contains over 600 such exercises. Answers to the odd-numbered section exercises are given in the back of the book, and answers to the even-numbered section exercises are provided on the instructor CD-ROM.
- Figures to facilitate the learning process.

Examples

The book contains 270 numbered examples, which clarify particular facets of Java. Most numbered examples are full programs or class definitions. A box ■ marks the end of each example.

Exercises

The book contains over 600 section review exercises, the answers to which are short answers, code segments, and, in a few cases, entire programs. These exercises are suitable as homework problems or as self-tests. The answers to the odd-numbered exercises are given in the back of the book. The Web site includes additional materials for self-study. My experience teaching Java has convinced me of the importance of these exercises.

Student and Instructor Support Materials

The book includes a CD-ROM with the JBuilder Integrated Development Environment for Java and the source code for all of the sample applications and major examples. The Java source code is also available at

`http://condor.depaul.edu/~mkalin`

The Instructor CD-ROM and Web Site provide full instructional support for courses using the text. The Instructor CD-ROM includes

- Solutions to even-numbered Section Exercises.
- Source code for the sample applications and main programs.
- Test suites consisting of multiple-choice questions for each chapter. There are over 160 multiple choice questions.
- Chapter outlines as Power Point slides.
- A sample syllabus.

The Web site's instructional support includes true/false review questions for each chapter. There are over 100 such questions. Source code for all of the sample applications and main programs also is available at the Web site.

Acknowledgments

I wish to thank the following reviewers for their generous help: Sergio Antoy, Portland State University; Chaya Gurwitz, CUNY Brooklyn College; Rex Jaeschke, independent consultant; and Celia Schahczenski, Montana Tech of the University of Montana.

I am indebted to the School of Computer Science, Telecommunications, and Information Systems at DePaul University and its dean, Helmut Epp, for encouraging the development of this book.

Once again I am grateful to Patricia Johnsonbaugh for her patient and insightful copy editing.

I received consistent support from the people at Prentice Hall. Special thanks go to Petra Recter, Senior Acquisitions Editor; Jennie Burger, Senior Marketing Manager; Sarah Burrows, Assistant Editor; and Irwin Zucker, Production Coordinator.

M.K.