Preface

Welcome to Java! At Deitel & Associates, we write college-level programming-language textbooks and professional books and work hard to keep our books up-to-date. Writing *Java How to Program, Fifth Edition,* (*5/e* for short), was a joy. This book and its support materials have everything instructors and students need for an informative, interesting, challenging and entertaining Java educational experience. As the book goes to publication, it is compliant with the latest version of Java—the *Java 2 Platform, Standard Edition (J2SE), version 1.4.1*—and with object-oriented design using the latest version of the *UML (Unified Modeling Language)* from the Object Management Group (OMG). We tuned the writing, the pedagogy, our coding style, the book's ancillary package and added a substantial treatment of developing database-driven Internet- and Web-based applications. We moved the *Tour of the Book* to the Preface. The tour will help instructors, students and professionals get a sense of the rich coverage the book provides of Java object-oriented programming, object-oriented design with the UML, and developing Internet- and Web-based applications. If you are evaluating the book, please be sure to read the *Tour of the Book*, which starts on page xxxvi.

Whether you are an instructor, a student, an experienced professional or a novice programmer, this book has much to offer. Java is a world-class programming language for developing industrial-strength computer applications for devices ranging from cell phones and PDAs to the largest enterprise servers. We carefully audited the manuscript against the *Java Language Specification*,[1] which defines Java. As a result, the programs you create by studying this text should work with any J2SE 1.4.1 compatible Java platform.

In this Preface, we overview *Java How to Program, 5/e*'s comprehensive suite of educational materials that help instructors maximize their students' Java learning experience. We explain conventions we use, such as syntax coloring the code examples, "code washing" and highlighting important code segments to help focus students' attention on the key con-

---

1. Electronic HTML and PDF copies of the *Java Language Specification* are available free at the Sun Microsystem's Java Web site at `java.sun.com/docs/books/jls/index.html`.

cepts introduced in each chapter. We overview the new features of *Java How to Program, 5/e*, including our enhanced treatment of object-oriented programming, Web-application development with servlets and JSP, the enhanced optional elevator-simulation object-oriented design (OOD) case study with the UML, the overview of design patterns and the extensive use of UML diagrams that have been upgraded to UML version 1.4 standards.

Prentice Hall has bundled a CD with the text that contains Sun Microsystem's *J2SE 1.4.1 Software Development Kit* (J2SDK) and their *Sun ONE Studio 4 (Community Edition)*, integrated development environment (IDE). To further support novice programmers, we offer several free *DIVE-INTO™ Series* publications that explain how to compile, execute and debug Java programs using the J2SDK, Sun ONE Studio (Community Edition) and Borland's *JBuilder Personal* edition. These publications are located at `www.deitel.com/books/downloads.html` with the resources for *Java How to Program, 5/e*.

We overview the complete package of ancillary materials available to instructors and students using *Java How to Program, 5/e*. These include an *Instructor's Resource CD* with solutions to most of the book's chapter exercises and a *Test-Item File* with hundreds of multiple-choice questions and answers. Additional instructor resources are available at the book's Companion Web Site (`www.prenhall.com/deitel`), which includes a *Syllabus Manager* and customizable PowerPoint® Lecture Notes. Numerous support materials are available for students at the Companion Web Site, as well. For instructors who want to hold closed-lab sessions (or highly structured homework assignments), we provide a lab manual, *Java in the Lab, Lab Manual to Accompany Java How to Program, Fifth Edition*. This publication includes carefully constructed Prelab Activities, Lab Exercises and Postlab Activities for a closed lab setting. Instructors can obtain the solutions manual to *Java in the Lab* from their regular Prentice Hall representatives.

We overview *The Java 2 Multimedia Cyber Classroom, 5/e*—an interactive, multimedia CD-based version of the book. This learning aid provides extensive interactivity features including hyperlinking, text search, audio "walkthroughs" of programs, Flash® animations and hundreds of exercises and solutions. We describe how to order both the *Cyber Classroom* and *The Complete Java 2 Training Course, 5/e*, boxed product, which contains the *Cyber Classroom* and the textbook later in the preface.

We discuss several DEITEL™ e-learning initiatives, including an explanation of Deitel content available for the *Blackboard*, *CourseCompass* and *WebCT* Course Management Systems, each of which supports *Java How to Program, 5/e*. *Premium CourseCompass*, which offers enhanced Deitel content based on *The Java 2 Multimedia Cyber Classroom, 5/e*, will be available for Summer 2003 courses.

In preparation for this edition, *Java How to Program, 4/e*, was reviewed by 35 distinguished academics and industry professionals. After applying their comments, the manuscript for *Java How to Program, 5/e*, was reviewed by 44 distinguished academics and industry professionals. We list all the reviewers names and affiliations in the acknowledgements. The Preface concludes with information about the authors and about Deitel & Associates, Inc. Please send an e-mail to `deitel@deitel.com` if you have questions as you read this book; we will respond promptly. Please visit our Web site, `www.deitel.com`, regularly and be sure to sign up for the *DEITEL™ BUZZ ONLINE* e-mail newsletter at `www.deitel.com/newsletter/subscribe.html`. We use the Web site and the newsletter to keep our readers current on *Java How to Program, 5/e*, and all other DEITEL™ publications and services.

New Features in Java How to Program, Fifth Edition

This edition contains many new features and enhancements including:

Full-Color Presentation

This book is in full color to show programs and their outputs as they typically appear on a computer screen. We syntax color all the Java code, as do most Java integrated-development environments and code editors. This greatly improves code readability—an especially important goal, given that this book contains over 23,000 lines of code. Our syntax-coloring conventions are as follows:

```
comments appear in green
keywords appear in dark blue
errors and JSP scriptlet delimiters appear in red
constants and literal values appear in light blue
all other code appears in black
```

Code Highlighting

We have added extensive code highlighting. In our code walkthroughs, we have eliminated most of the "redundant" code snippets that appeared inline in the text in earlier editions. We kept them in the earliest portion of the book as a pedagogic device to help novices. We want the reader to see all new code features in context, so from Chapter 4 forward, our code walkthroughs simply refer to the line numbers of the new code segments inside complete source programs. To make it easier for readers to spot the featured segments, we highlight them in bright yellow. This helps students review the material rapidly when preparing for exams or labs.

"Code Washing"

Code washing is our term for applying extensive comments, using meaningful identifiers, applying indentation and using vertical spacing to separate meaningful program units. This process results in programs that are much more readable and self-documenting. We have done extensive "code washing" of all the source code programs in the text, the lab manual, the ancillaries and the *Cyber Classroom.*

Tuned Treatment of Object-Oriented Programming in Chapters 9 and 10

This is one of the most significant improvements in this new edition. We performed a high-precision upgrade of *Java How to Program, 4/e*, Chapter 9 and split it into two chapters. The improvements make the material clearer and more accessible to students and professionals, especially those studying object-oriented programming for the first time.

 Chapter 9, Object-Oriented Programming: Inheritance. The new Chapter 9 carefully walks the reader through a five-example sequence that demonstrates `private` data, `protected` data and software reuse via inheritance. We begin by demonstrating a class with `private` instance variables and `public` methods to manipulate that data. Next, we implement a second class with several additional capabilities. To do this, we duplicate much of the first example's code. In our third example, we begin our discussion of inheritance and software reuse—we use the class from the first example as a superclass and inherit its data and functionality into a new subclass. This example introduces the inheritance mechanism and demonstrates that a subclass cannot access its superclass's `private` members directly. This motivates our fourth example, in which we introduce `protected` data in the super-

class and demonstrate that the subclass can indeed access the `protected` data inherited from the superclass. The last example in the sequence demonstrates proper software engineering by defining the superclass's data as `private` and using the superclass's `public` methods (that were inherited by the subclass) to manipulate the superclass's `private` data from the subclass. We follow the five-part introduction with a three-level class hierarchy that employs the software engineering techniques introduced earlier in the chapter. The chapter closes with a discussion of software engineering with inheritance.

    Chapter 10, Object-Oriented Programming: Polymorphism. The new Chapter 10 builds on the inheritance concepts presented in Chapter 9 and focuses on the relationships among classes in a class hierarchy. Chapter 10 uses a three-example sequence to present the powerful processing capabilities that these relationships enable. We begin with an example that illustrates the "is-a" relationship between a subclass object and its superclass type. This relationship enables the subclass object to be treated as an object of its superclass. We show that we are able to assign a subclass object's reference to a superclass variable and invoke the superclass's methods on that object. This example uses polymorphism, which enables a program to process objects of classes related by a class hierarchy as objects of their superclass type. When a method is invoked via a superclass variable, the subclass-specific version of that method is invoked. In our second example, we demonstrate that the reverse is not true—a superclass object is not considered to be an object of its subclass type—and we show that compiler errors occur if a program attempts to manipulate a superclass object in this manner. Our third example demonstrates that the only methods which can be invoked through a superclass variable are those methods defined by the superclass type. The example shows that attempts to invoke subclass-only methods result in compilation errors. The chapter continues with a case study on polymorphism in which we process an array of variables that contain references to objects. All the objects referenced by the elements of the array have a common abstract superclass containing the set of methods common to every class in the hierarchy. We conclude with a case study that demonstrates how a program that processes objects polymorphically can still perform type-specific processing by determining the type of the object currently being processed.

Java New I/O (NIO) APIs

Java's New I/O APIs are significant new additions to J2SE 1.4. We overview portions of these APIs in sections of three chapters. Section 11.8 demonstrates NIO's regular expression capabilities, which enable programs to search strings for character patterns. Section 17.13 introduces NIO's high-performance I/O classes that enable developers to take advantage of buffers, channels, charsets and more. This section also presents an example of using channels and buffers to write data to, and read data from, a file. Section 18.11 continues our discussion of the NIO APIs with an introduction to selectors and non-blocking I/O for implementing high-performance network servers. We then implement a distributed chat program that demonstrates these capabilities. Sections 11.8 and 17.13 also provide Web links for further study of the NIO APIs.

Database and Web-Applications Development with JDBC, Servlets and JSP

By popular demand, we have returned several topics to *Java How to Program, Fifth Edition*. Chapter 23, Java Database Connectivity with JDBC, demonstrates how to build data-driven applications with the JDBC™ API. Chapter 24, Servlets, and Chapter 25, JavaServer Pages™ (JSP), expand our treatment of Internet and Web programming topics and have

everything readers need to begin developing their own Web-based applications that will run on the Internet! Readers will learn how to build so-called *n*-tier applications, in which the functionality provided by each tier can be distributed to separate computers across the Internet or executed on the same computer. In particular, we build a three-tier Web-based survey application and a three-tier Web-based guestbook application. Each application's information is stored in the application's data tier—in this book, a database implemented with IBM's Java-based Cloudscape database product (a trial version is on the CD that accompanies this book). The user enters requests and receives responses at each application's client tier, which is typically a computer running a Web browser such as Microsoft Internet Explorer or Netscape. Web browsers, of course, know how to communicate with Web sites throughout the Internet. The middle tier contains both a Web server and one or more application-specific servlets (in the case of our survey application) or JavaServer Pages (in the case of our guestbook application). We use Apache's Tomcat Web server as our application server for these examples. Tomcat, which is the reference implementation for the servlets and JavaServer Pages technologies, is included on the CD that accompanies this book and is available free for download from `www.apache.org`. Tomcat communicates with the client tier across the Internet using the HyperText Transfer Protocol (HTTP). We discuss the crucial role of the Web server in Web programming and provide many examples demonstrating interactions between a Web browser and a Web server.

Unified Modeling Language™ (UML)

The Unified Modeling Language™ (UML) has become the preferred graphical modeling language for designing object-oriented systems. In *Java How to Program, Fourth Edition*, we used the UML in optional sections only, and we used conventional flowchart segments and inheritance diagrams to reinforce the explanations. We have fully converted the diagrams in the book to be UML 1.4 compliant. In particular, we upgraded all the figures in the UML/OOD Elevator Simulation case study; we converted all the flowcharts in Chapters 4 and 5 on Control Statements, to UML activity diagrams; and we converted all the inheritance diagrams in Chapters 9, 10, 12–13 and 15 to UML class diagrams.

This *Fifth Edition* carefully tunes the optional (but highly recommended) case study we present on object-oriented design using the UML. The case study was submitted to a distinguished team of OOD/UML reviewers, including leaders in the field from Rational (the creators of the UML) and the Object Management Group (responsible for maintaining and evolving the UML). In the case study, we fully implement an elevator simulation. In the "Thinking About Objects" sections at the ends of Chapters 1–8, 10–14, 16 and 19, we present a carefully paced introduction to object-oriented design using the UML. We present a concise, simplified subset of the UML then guide the reader through a first design experience intended for the novice object-oriented designer/programmer. The case study is fully solved. It is not an exercise; rather, it is an end-to-end learning experience that concludes with a detailed walkthrough of the Java code. In each of the first five chapters, we concentrate on the "conventional" methodology of structured programming, because the objects that we build will use these structured-program pieces. We conclude each chapter with a "Thinking About Objects" section, in which we present an introduction to object-oriented design (OOD) using the UML. These "Thinking About Objects" sections help students develop an object-oriented design, so that they immediately can use the object-oriented programming concepts they begin learning in Chapter 8. In the first of these sections at the end of Chapter 1, we introduce basic concepts and terminology of OOD. In the optional

"Thinking About Objects" sections at the ends of Chapters 2–5, we consider more substantial issues, as we undertake a challenging problem with the techniques of OOD. We analyze a typical problem statement that requires a system to be built, determine the objects needed to implement that system, determine the attributes these objects need to have, determine the behaviors these objects need to exhibit and specify how the objects need to interact with one another to meet the system requirements. We accomplish this even before we discuss how to write Java programs. In Appendices D–F, we include a Java implementation of the object-oriented system we designed in the earlier chapters. This case study will help prepare students for the kinds of substantial projects they will encounter in industry. We employ a carefully developed, incremental object-oriented design process to produce a UML model for our elevator simulator. From this design, we produce a substantial working Java implementation using key programming notions, including classes, objects, encapsulation, visibility, composition and inheritance.

Discovering Design Patterns

These optional sections introduce popular object-oriented design patterns. Over the past decade, the software engineering industry has made significant progress in the field of *design patterns*—proven architectures for constructing flexible and maintainable object-oriented software.[2] Using design patterns can substantially reduce the complexity of the design process. We present several design patterns in Java, but these can be implemented in any object-oriented language, such as C++, C# or Visual Basic .NET. We describe several design patterns used by Sun Microsystems in the Java API. We use design patterns in many programs in this book, which we will identify in our "Discovering Design Patterns" sections. These programs provide examples of using design patterns to construct reliable, robust object-oriented software.

Teaching Approach

Java How to Program, Fifth Edition contains a rich collection of examples, exercises, and projects drawn from many fields to provide the student with a chance to solve interesting real-world problems. The book concentrates on the principles of good software engineering and stresses program clarity. We avoid arcane terminology and syntax specifications in favor of teaching by example. Our code examples have been tested on popular Java platforms. We are educators who teach edge-of-the-practice topics in industry classrooms worldwide. The text emphasizes good pedagogy.

Learning Java via the LIVE-CODE™ Approach

Java How to Program, 5/e, is loaded with LIVE-CODE™ examples. Each new concept is presented in the context of a complete, working example that is immediately followed by one or more sample executions showing the program's input/output dialog. This style exemplifies the way we teach and write about programming and is the focus of our multimedia *Cyber Classrooms* and Web-based training courses. We call this method of teaching and writing the **LIVE-CODE™ Approach**. *We use programming languages to teach pro-*

---

2. Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns; Elements of Reusable Object-Oriented Software*. (Massachusetts: Addison-Wesley, 1995).

gramming languages. Reading the examples in the text is much like typing and running them on a computer. We provide all the source code for the book's examples on both the accompanying CD and at `www.deitel.com`. We encourage you to run every example.

Java Programming with Applications and Swing from Chapter Two!

Java How to Program, 5/e, "jumps right into" programming Java applications with the Swing GUI components from Chapter 2. There is great stuff to be done in Java so let's get right to it! Java is not trivial by any means, but it's fun to program with and students can see immediate results. Students can get graphical, animated, multimedia-based, audio-intensive, multithreaded, database-intensive, network-based programs running quickly through Java's extensive class libraries of reusable components. They can implement impressive projects. They are typically more creative and productive in a one- or two-semester course than in C and C++ introductory courses.

World Wide Web Access

All of the source-code examples for *Java How to Program, 5/e,* (and our other publications) are available on the Internet as downloads from the following Web sites:

```
www.deitel.com
www.prenhall.com/deitel
```

Registration is quick and easy and the downloads are free. We suggest downloading all the examples, then running each program as you read the corresponding text. Making changes to the examples and immediately seeing the effects of those changes is a great way to enhance your Java learning experience.

Objectives

Each chapter begins with objectives that inform students of what to expect and give them an opportunity, after reading the chapter, to determine whether they have met the intended objectives. The objectives serve as confidence builders.

Quotations

The chapter objectives are followed by sets of quotations. Some are humorous, some are philosophical and some offer interesting insights. We have found that students enjoy relating the quotations to the chapter material. Many of the quotations are worth a second look *after* you read the chapters.

Outline

The chapter outline enables students to approach the material in a top-down fashion. Along with the chapter objectives, the outline helps students anticipate future topics and set a comfortable and effective learning pace.

23,341 Lines of Code in 219 Example Programs (with Program Outputs)

We present Java features in the context of complete, working Java programs. These LIVE-CODE™ programs range in size from just a few lines of code to substantial examples containing hundreds of lines of code. Each program is followed by a window containing the outputs produced when the program is run, so students can confirm that the programs run as expected. Relating outputs back to the program statements that produce those outputs is

an excellent way to learn and to reinforce concepts. Our programs exercise the diverse features of Java. The code is syntax colored with Java keywords, comments and other program text each appearing in different colors. This facilitates reading the code—students especially will appreciate the syntax coloring when they read the larger programs we present.

615 Illustrations/Figures

An abundance of charts, line drawings, programs and program outputs is included. We have converted all flowcharts to UML activity diagrams. We also use UML class diagrams to model the relationships between classes throughout the text.

534 Programming Tips

We have included programming tips to help students focus on important aspects of program development. We highlight hundreds of these tips in the form of *Good Programming Practices*, *Common Programming Errors*, *Error-Prevention Tips, Look-and-Feel Observations*, *Performance Tips*, *Portability Tips* and *Software Engineering Observations*. These tips and practices represent the best we have gleaned from a combined six decades of programming and teaching experience. One of our students—a mathematics major—told us that she feels this approach is like the highlighting of axioms, theorems, and corollaries in mathematics books; it provides a basis on which to build good software.

82 Good Programming Practices

Good Programming Practices *are tips that call attention to techniques for writing clear programs. These techniques help students produce programs that are more readable, self-documenting and easier to maintain.*

156 Common Programming Errors

Students learning a language—especially in their first programming course—tend to make certain kinds of errors frequently. Focusing on these Common Programming Errors *reduces the likelihood that students will makes the same mistakes. It also shortens long lines outside instructors' offices during office hours!*

50 Error-Prevention Tips

When we first designed this "tip type," we thought we would use it strictly to tell people how to test and debug Java programs. In fact, many of the tips describe aspects of Java that reduce the likelihood of "bugs" and thus simplify the testing and debugging processes.

36 Look-and-Feel Observations

We provide Look-and-Feel Observations *to highlight graphical user interface conventions. These observations help students design their own graphical user interfaces in conformance with industry norms.*

52 Performance Tips

In our experience, teaching students to write clear and understandable programs is by far the most important goal for a first programming course. But students want to write the programs that run the fastest, use the least memory, require the smallest number of keystrokes, or dazzle in other nifty ways. Students really care about performance. They want to know what they can do to "turbo charge" their programs. So we highlight opportunities for improving program performance—making programs run faster or minimizing the amount of memory that they occupy.

23 Portability Tips

One of Java's "claims to fame" is "universal" portability, so some programmers assume that if they implement an application in Java, the application will automatically be "perfectly" portable across all Java platforms. Unfortunately, this is not always the case. We include Portability Tips *to help students write portable code and to provide insights on how Java achieves its high degree of portability. We had many more portability tips in our books,* C How to Program *and* C++ How to Program. Java How to Program *has fewer of these tips because Java is designed to be portable top-to-bottom (for the most part)—much less effort is required on the Java programmer's part to achieve portability than with C or C++.*

135 Software Engineering Observations

The object-oriented programming paradigm requires a complete rethinking about the way we build software. Java is an effective language for performing good software engineering. The Software Engineering Observations *highlight architectural and design issues that affect the construction of software systems, especially large-scale systems. Much of what the student learns here will be useful in upper-level courses and in industry as the student begins to work with large, complex real-world systems.*

Summary (954 Summary bullets)

Each chapter ends with additional pedagogical devices. We present a thorough, bullet-list-style summary of the chapter. On average, there are 38 summary bullets per chapter. This helps the students review and reinforce key concepts.

Terminology (2166 Terms)

We include in a *Terminology* section an alphabetized list of the important terms defined in the chapter—again, further reinforcement. On average, there are 87 terms per chapter.

437 Self-Review Exercises and Answers (Count Includes Separate Parts)

Extensive self-review exercises and answers are included for self-study. This gives the student a chance to build confidence with the material and prepare for the regular exercises. Students should be encouraged to do all the self-review exercises and check their answers.

858 Exercises (Count Includes Separate Parts)

Each chapter concludes with a set of exercises, including simple recall of important terminology and concepts; writing individual Java statements; writing small portions of Java methods and classes; writing complete Java methods, classes, applications and applets; and writing major term projects. The large number of exercises across a wide variety of areas enables instructors to tailor their courses to the unique needs of their audiences and to vary course assignments each semester. Instructors can use these exercises to form homework assignments, short quizzes and major examinations. The solutions for most of the exercises are included on the *Instructor's Resource CD,* which is *available only to instructors* through their Prentice Hall representatives. **[NOTE: Please do not write to us requesting the Instructor's CD. Distribution of this ancillary is limited strictly to college professors teaching from the book. Instructors may obtain the solutions manual only from their Prentice Hall representatives.]** Students and professional readers can obtain solutions to approximately half the exercises in the book by purchasing the optional *Java 2 Multimedia Cyber Classroom, 5/e*. The *Cyber Classroom* offers many other features and is ideal for self study and reference. Also available is the boxed product, *The Complete Java 2 Training*

Course, 5/e, which includes both our textbook, *Java How to Program, 5/e*, and the *Java 2 Multimedia Cyber Classroom, 5/e*. All of our *Complete Training Course* products are available at bookstores and online booksellers, including `www.informIT.com`. If you already have the textbook, you can purchase the *Java 2 Multimedia Cyber Classroom, 5/e* (ISBN# 0-13-101769-1), separately at `www.InformIT.com/cyberclassrooms`.

Approximately 4800 Index Entries (with approximately 8000 Page References)

We have included an extensive *Index* at the back of the book. Using this resource, readers can search for any term or concept by keyword. The *Index* is useful to people reading the book for the first time and is especially useful to professional programmers who use the book as a reference. These index entries also appear as hyperlinks in the *Java 2 Multimedia Cyber Classroom, 5/e*.

"Double Indexing" of Java LIVE-CODE™ Examples

We have "double indexed" *Java How to Program*'s 219 LIVE-CODE™ examples. For every Java source-code program in the book, we took the figure caption and indexed it both alphabetically and as a subindex item under "Examples." This makes it easier to find examples using particular features.

Bibliography

An extensive bibliography of books, articles and Sun Microsystems Java 2 documentation is included to encourage further reading.

Software Included with Java How to Program, Fifth Edition

There are a number of for-sale Java development tools available. However, you do not need them to get started with Java. We wrote *Java How to Program, 5/e*, using only the *Java 2 Standard Edition Software Development Kit (J2SDK), version 1.4.1.* For your convenience, Sun's J2SDK 1.4.1 is included on the CD that accompanies this book. The current J2SDK version can always be downloaded from Sun's Java Web site `java.sun.com/j2se`. This site also contains the J2SDK documentation downloads.

With Sun's cooperation, we also were able to include on the CD a powerful Java integrated development environment (IDE)—*Sun ONE™ Studio 4, Community Edition*. *Sun ONE™ Studio 4, Community Edition*, is a professional IDE written in Java that includes a graphical user interface designer, code editor, compiler, visual debugger and more. The J2SDK must be installed before installing *Sun ONE™ Studio 4 Community Edition*. If you have any questions about using this software, please read the documentation on the CD, or read our *DIVE-INTO™ Series* publication *Dive Into Sun ONE Studio 4, Community Edition*. This document is available with the resources for *Java How to Program, 5/e*, at `www.deitel.com/books/downloads.html`.

The CD contains the book's examples (including the Elevator Case Study implementation) and an HTML Web page with links to the Deitel & Associates, Inc. Web site and the Prentice Hall Web site. If you have access to the Internet, this Web page can be loaded into your Web browser to give you quick access to all the resources. In addition, we provide several chapters and appendices from other Deitel publications. These include material on XHTML and Cascading Style Sheets (for use with Chapter 24, Servlets, and Chapter 25, JavaServer Pages), and material on Extensible Markup Language (XML) and Java's XML-processing APIs, which are now part of J2SE 1.4.

Ancillary Package for Java How to Program, Fifth Edition

Java How to Program, 5/e, has extensive ancillary materials for instructors. The *Instructor's Resource CD (IRCD)* contains the *Instructor's Manual* with solutions to the vast majority of the end-of-chapter exercises and a *Test Item File* of multiple-choice questions (approximately two per book section). In addition, we provide PowerPoint slides containing all the code and figures in the text, and bulleted items that summarize the key points in the text. Instructors can customize the slides. The PowerPoint slides are downloadable from `www.deitel.com` and are available as part of Prentice Hall's *Companion Web Site* (`www.prenhall.com/deitel`) for *Java How to Program, 5/e*, which offers resources for both instructors and students. For instructors, the *Companion Web Site* offers a *Syllabus Manager*, which helps instructors plan courses interactively and create online syllabi.

Students also benefit from the functionality of the *Companion Web Site*. Book-specific resources for students include:

- Customizable PowerPoint® slides
- Example source code
- Reference materials from the book appendices (such as operator-precedence chart, character set and Web resources)

Chapter-specific resources available for students include:

- Chapter objectives
- Highlights (e.g., chapter summary)
- Outline
- Tips (e.g., *Common Programming Errors*, *Error-Prevention Tips, Good Programming Practices*, *Look-and-Feel Observations*, *Portability Tips*, *Performance Tips* and *Software Engineering Observations*)
- Online Study Guide—contains additional short-answer self-review exercises (e.g., true/false) with answers and provides immediate feedback to the student

Students can track their results and course performance on quizzes using the *Student Profile* feature, which records and manages all feedback and results from tests taken on the *Companion Web Site*. To access the *Companion Web Site*, visit `www.prenhall.com/deitel`.

Java in the Lab

This lab manual (full title: *Java in the Lab, Lab Manual to Accompany Java How to Program, 5/e*; ISBN# is 0-13-101631-8[3]) complements *Java How to Program, 5/e,* and the optional *Java 2 Multimedia Cyber Classroom, 5/e,* with hands-on lab assignments designed to reinforce students' understanding of lecture material. This lab manual is designed for closed laboratories, which are regularly scheduled classes supervised by an instructor. Closed laboratories provide an excellent learning environment because students can use concepts presented in class to solve carefully designed lab problems. Instructors are better able to gauge the students' understanding of the material by monitoring the students' progress in lab. This lab manual also can be used for open laboratories, homework and for self-study.

---

3. *Java How to Program, 5/e*, and the lab manual also are available together in a value pack (ISBN# 0-13-102719-0).

Java in the Lab focuses on Chapters 1–12, 15 and 17 of *Java How to Program, 5/e*. Each chapter in the lab manual is divided into *Prelab Activities*, *Lab Exercises* and *Postlab Activities*.[4] Each chapter contains objectives that introduce the lab's key topics and an assignment checklist that allows students to mark which exercises the instructor has assigned. The lab manual pages are perforated, so students can submit their answers (if required).

Solutions to the lab manual's *Prelab Activities*, *Lab Exercises* and *Postlab Activities* are available in electronic form. Instructors can obtain these materials from their regular Prentice Hall representatives; the solutions are not available to students.

Prelab Activities

Prelab Activities are intended to be completed by students after studying each chapter in *Java How to Program, 5/e*. *Prelab Activities* test students' understanding of the material presented in the textbook, and prepare students for the programming exercises in the lab session. The exercises focus on important terminology and programming concepts and are effective for self-review. Prelab Activities include *Matching Exercises*, *Fill-in-the-Blank Exercises*, *Short-Answer Questions*, *Programming-Output Exercises* (determine what short code segments do without actually running the program) and *Correct-the-Code Exercises* (identify and correct all errors in short code segments).

Lab Exercises

The most important section in each chapter is the Lab Exercises. These exercises teach students how to apply the material learned in *Java How to Program, 5/e*, and prepare them for writing Java programs. Each lab contains one or more lab exercises and a debugging problem. The *Lab Exercises* contain the following:

- *Lab Objectives* highlight specific concepts on which the lab exercise focuses.

- *Problem Descriptions* provide the details of the exercise and hints to help students implement the program.

- *Sample Outputs* illustrate the desired program behavior, which further clarifies the problem descriptions and aids the students with writing programs.

- *Program Templates* take complete Java programs and replace key lines of code with comments describing the missing code.

- *Problem-Solving Tips* highlight key issues that students need to consider when solving the lab exercises.

- *Follow-Up Questions and Activities* ask students to modify solutions to lab exercises, write new programs that are similar to their lab-exercise solutions or explain the implementation choices that were made when solving lab exercises.

- *Debugging Problems* consist of blocks of code that contain syntax errors and/or logic errors. These alert students to the types of errors they are likely to encounter while programming.

---

4. We expect few introductory classes to advance beyond Chapter 11 of this lab manual. For this reason, the labs in Chapters 12, 15 and 17 do not contain the extensive sets of activities available in the previous chapters. Nevertheless, instructors will be able to conduct effective labs using the exercises we have included on these more complex topics. Instructors with special requirements should write to `deitel@deitel.com`.

Postlab Activities

Professors typically assign *Postlab Activities* to reinforce key concepts or to provide students with more programming experience outside the lab. *Postlab Activities* test the students' understanding of the *Prelab* and *Lab Exercise* material, and ask students to apply the knowledge to creating programs from scratch. The section provides two types of programming activities: coding exercises and programming challenges. Coding exercises are short and serve as review after the *Prelab Activities* and *Lab Exercises* have been completed. These ask students to write programs or program segments using key concepts from the textbook. *Programming Challenges* allow students to apply the knowledge they have gained in class to substantial programming exercises. Hints, sample outputs and/or pseudocode are provided to aid students with these problems. Students who complete the *Programming Challenges* for a chapter successfully have mastered the chapter material. Answers to the programming challenges are available at `www.deitel.com/books/downloads.html`.

Java 2 Multimedia Cyber Classroom, 5/e, and *The Complete Java 2 Training Course, 5/e*

We have updated our optional interactive multimedia version of the book—*The Java 2 Multimedia Cyber Classroom, 5/e* (CD for Windows®)—with considerable additional audio, including the new material on database development with JDBC and Web-applications development with servlets and JavaServer Pages. This resource is loaded with electronic learning and reference features. The *Cyber Classroom* is packaged with the textbook at a discount in *The Complete Java 2 Training Course, 5/e* (ISBN# 0-13-101766-7). If you already have the book and would like to purchase the *Cyber Classroom* separately, please visit `www.InformIT.com/cyberclassrooms`; the ISBN number for the *Cyber Classroom* is 0-13-101769-1. Deitel™ *Cyber Classrooms* are generally available in CD and various popular Web-based training formats.

The CD provides an introduction in which the authors overview the *Cyber Classroom*'s features. The textbook's 219 LIVE-CODE™ example Java programs truly "come alive" in the *Cyber Classroom*. When viewing a program, simply click the lightning-bolt icon to run the program. You will immediately see the program's output. If you want to modify a program and see the effects of your changes, simply clicking the floppy-disk icon causes the source code to be "lifted off" the CD and "dropped into" one of your own directories so you can edit the code, recompile the program and run your new version. Click the audio icon to hear one of the authors "walk you through" the code. In addition, the *Cyber Classroom* contains the full-text of *Java How to Program, 5/e*, in fully-searchable format.

The *Cyber Classroom* also provides post-assessment exams (with answers) for each chapter in the book. These exams are powerful features that allow users to gauge their understanding of the programming concepts presented in the chapters. Each exam question hyperlinks to the section in the book from which the question was derived. This allows users to review the appropriate chapter material before or after answering the question. A chart is provided that summarizes the user's exam results by chapter.

The *Cyber Classroom* also provides navigational aids, including extensive additional hyperlinking for easy navigation. The *Cyber Classroom* is browser based, so it remembers sections that you have visited recently and allows you to move forward or backward among them. The thousands of index entries are hyperlinked to their text occurrences. You can use

the "find" feature to locate occurrences of a term throughout the text. The Table of Contents entries are "hot," so clicking a chapter or section name takes you immediately to that chapter or section.

Students like the fact that solutions to approximately half the exercises in the book are included with the *Cyber Classroom*. Studying and running these extra programs is a nice way for students to enhance their LIVE-CODE™ learning experience.

Students and professional users of our *Cyber Classrooms* tell us that they like the interactivity and that the *Cyber Classroom* is a powerful reference tool. We received an e-mail from a person who said that he lives "in the boonies" and cannot take a live course at a university, so the *Cyber Classroom* provided a nice solution to his educational needs.

Professors tell us that their students enjoy using the *Cyber Classroom*, and consequently spend more time on the courses, mastering more of the material than in textbook-only courses. For a complete list of the available and forthcoming *Cyber Classrooms* and *Complete Training Courses*, see the *Deitel™ Series* page at the beginning of this book, the product listing and ordering information at the end of this book or visit `www.deitel.com`, `www.prenhall.com/deitel` or `www.InformIT.com/deitel`.

Advanced Java™ 2 Platform How to Program

Our companion book—***Advanced Java 2 Platform How to Program***—focuses on the ***Java 2 Platform, Enterprise Edition (J2EE)***, presents advanced Java 2 Platform Standard Edition features and introduces the ***Java 2 Platform, Micro Edition (J2ME)***. This book is intended for developers and upper-level university students in advanced courses who already know Java and want a deeper treatment and understanding of the language. The book features our signature LIVE-CODE™ approach of complete working programs and contains over 37,000 lines of code. The programs are more substantial than those presented in *Java How to Program, Fifth Edition*. The book expands the coverage of Java Database Connectivity (JDBC), servlets and JavaServer Pages (JSP) from *Java How to Program, Fifth Edition*. The book also covers emerging and more advanced Java technologies of concern to enterprise application developers, including Model-View-Controller; Java 2D and Java 3D; JavaBeans Component Model; Security; Java 2 Micro Edition (J2ME) and Wireless Internet; Remote Method Invocation (RMI); Enterprise JavaBeans (EJBs); Java Message Service (JMS); Jini; JavaSpaces; Jiro; Java Management Extensions (JMX); Common Object Request Broker Architecture (CORBA); Peer-to-Peer Networking; Web Services; XML and Java Native Interface (JNI).

Java Web Services for Experienced Programmers

Part of the new *Deitel Developer Series* for computer professionals, Java *Web Services for Experienced Programmers* uses our proven LIVE-CODE™ approach to teach the latest XML and Java technologies for building and integrating Web services, including the *Java Web Services Developer Pack*. This book is designed for industry professionals who require in-depth coverage of Java Web-services technologies. The book is also suitable for upper-level computer science courses in Web services or as a supplement to courses in distributed computing and advanced Java programming. Instructors should contact their Prentice Hall representatives to obtain examination copies. Topics covered include XML; Document Type Definitions (DTDs); Document Object Model (DOM™) and the Java API

for XML Processing (JAXP); eXtensible Stylesheets Transformations (XSLT™) and the Transformation API for XML (TrAX); Simple Object Access Protocol (SOAP); Web Services Description Language (WSDL); Universal Description, Discovery and Integration (UDDI) and the Java API for XML Registries (JAXR); Java API for XML-based Remote Procedure Calls (JAX-RPC); Java API for XML Messaging (JAXM) and the SOAP with Attachments API for Java (SAAJ); Web Services Security with Secure Sockets Layer (SSL), XML Signature, XML Encryption, XML Key Management Specification (XKMS), Security Assertions Markup Language (SAML) and eXtensible Access Control Markup Language (XACML); and Wireless Web Services with Java 2 Micro Edition (J2ME™).

Course Management Systems: Blackboard™, WebCT™, CourseCompass^SM and Premium CourseCompass^SM

Selected content from the Deitels' introductory programming language *How to Program* series, including *Java How to Program, 5/e*,[5] is available to integrate into various popular course management systems, including CourseCompass, Blackboard and WebCT. An enhanced version of CourseCompass, called Premium CourseCompass, will be available for *Java How to Program, 5/e,* for Summer 2003 courses. Course management systems help faculty create, manage and use sophisticated Web-based educational tools and programs. Instructors can save hours of inputting data by using Deitel course-management-systems content.

Blackboard, CourseCompass and WebCT offer:

- **Features to create and customize an online course**, such as areas to post course information (e.g., policies, syllabi, announcements, assignments, grades, performance evaluations and progress tracking), class and student management tools, a gradebook, reporting tools, page tracking, a calendar and assignments.

- **Communication tools** to help create and maintain interpersonal relationships between students and instructors, including chat rooms, whiteboards, document sharing, bulletin boards and private e-mail.

- **Flexible testing tools** that allow an instructor to create online quizzes and tests from questions directly linked to the text, and that grade and track results effectively. All tests can be inputted into the gradebook for efficient course management. WebCT also allows instructors to administer timed online quizzes.

- **Support materials** for instructors are available in print and online formats.

In addition to the types of tools found in Blackboard and WebCT, CourseCompass from Prentice Hall includes:

- **CourseCompass course home page**, which makes the course as easy to navigate as a book. An expandable table of contents allows instructors to view course content at a glance and to link to any section.

- **Hosting on Prentice Hall's centralized servers**, which allows course administrators to avoid separate licensing fees or server-space issues. Access to Prentice Hall technical support is available.

---

5. The entire text of *Java How to Program, 5/e,* is included in the e-Book included with Premium CourseCompass.

- **"How Do I" online-support sections** are available for users who need help personalizing course sites, including step-by-step instructions for adding Power-Point® slides, video and more.
- **Instructor Quick Start Guide** helps instructors create online courses using a simple, step-by-step process.

Premium CourseCompass Course Management System

Premium CourseCompass integrates content from several sources, including Deitel *Cyber Classrooms*, *How to Program* books and *Companion Web Sites* with CourseCompass courseware—providing enhanced content to CourseCompass users. Premium CourseCompass includes:

- **Pre-Loaded DEITEL™ Content in a Customizable Interface**. An instructor can aggregate and customize all course materials. This feature includes the e-Book, a searchable digital version of *Java How to Program, 5/e*, including full-color graphics and downloadable PowerPoint® slides.
- **All the Interactivity of the *Cyber Classroom*.** Students can work with code and receive the added benefit of 17+ hours of detailed audio descriptions of thousands of lines of code to help reinforce concepts. Every code example from *Java How to Program, 5/e*, is included.
- **Abundant Self-Assessment and Complete *Test-Item File*.** Use or edit hundreds of pre-loaded assessments, or upload your own. Assessments include self-review exercises, programming exercises (half with answers included) and test questions. Instructors choose which questions to assign, and students receive immediate feedback. Instructors can collect students' work and track their progress in an online gradebook.

To view free online demonstrations and learn more about these Course Management Systems, that support Deitel content, visit the following Web sites:

- Blackboard: `www.blackboard.com` and `www.prenhall.com/blackboard`
- WebCT: `www.webct.com` and `www.prenhall.com/webct`
- CourseCompass: `www.coursecompass.com` and
  `www.prenhall.com/coursecompass`

Computer Science AP Courses Switching to Java in Fall 2003

The AP Computer Science Program recently decided to move the AP computer science curriculum from C++ to Java starting with classes in the Fall of 2003. The first Java-based AP Computer-Science exams will be administered in the Spring of 2004. *Java How to Program, 5/e*, is a suitable textbook for instructors teaching AP Computer-Science classes and for preparing students to take the corresponding exams. While writing this book, we carefully reviewed the goals of the AP Computer Science A and AB exams, to ensure that *Java How to Program, 5/e,* covers the information required for the exams. At the time of this publication, the syllabi for the these exams had not yet been finalized. Instructors and students interested in preparing for these exams should visit:

`www.deitel.com/books/jHTP5/Java_AP_Exam.html`

dedicated to the use of *Java How to Program, 5/e*, in the Computer Science AP curriculum. We will update this site regularly with additional information about the exams. For detailed information on the Computer Science AP curriculum, please visit

```
apcentral.collegeboard.com
```

Deitel e-Learning Initiatives

e-Books and Support for Wireless Devices

Wireless devices will have an enormous role in the future of the Internet. Given recent bandwidth enhancements and the emergence of 2.5 and 3G technologies, it is projected that, within a few years, more people will access the Internet through wireless devices than through desktop computers. Deitel & Associates is committed to wireless accessibility and recently published *Wireless Internet & Mobile Business How to Program*. To fulfill the needs of a wide range of customers, we currently are developing our content both in traditional print formats and in newly developed electronic formats, such as wireless e-books so that students and professors can access content virtually anytime, anywhere. For periodic updates on these initiatives subscribe to the *Deitel™ Buzz Online* e-mail newsletter, `www.deitel.com/newsletter/subscribe.html` or visit `www.deitel.com`.

e-Matter

Deitel & Associates is partnering with Prentice Hall's parent company, Pearson PLC, and its information technology Web site, `www.InformIT.com`, to launch the DEITEL™ e-Matter series at `www.InformIT.com/deitel` in 2003. This series will provide professors, students and professionals with an additional source of information on programming and software topics. e-Matter consists of stand-alone sections taken from published texts, forthcoming texts or pieces written during the Deitel research-and-development process. Developing e-Matter based on pre-publication books allows us to offer significant amounts of the material to early adopters for use in academic and corporate courses.

Deitel and InformIT Newsletters

Deitel Newsletter

Our own free e-mail newsletter, the *DEITEL™ BUZZ ONLINE*, includes commentary on industry trends and developments, links to free articles and resources from our published books and upcoming publications, product-release schedules, challenges, anecdotes, information on our corporate instructor-led training courses and more. To subscribe, visit

```
www.deitel.com/newsletter/subscribe.html
```

Deitel Column in the InformIT Newsletters

Deitel & Associates, Inc., contributes to two free *InformIT* weekly e-mail newsletters, currently subscribed to by more than 1,000,000 IT professionals worldwide.

- *Editorial Newsletter*—Contains dozens of new articles per week on various IT topics, including programming, advanced computing, networking, business, Web development, software engineering, operating systems and more. Deitel & Associates contributes 2–3 articles per week taken from our extensive content base or from material being created during our research and development process.

- *Promotional Newsletter*—Features weekly specials and discounts on most Pearson publications. Each week a new DEITEL™ product is featured along with information about our corporate instructor-led training courses.

To subscribe, visit `www.InformIT.com`.

The New DEITEL™ Developer Series

Deitel & Associates, Inc., is making a major commitment to covering leading-edge technologies for industry software professionals through the launch of our *DEITEL™ Developer Series*. *Web Services A Technical Introduction* and *Java Web Services for Experienced Programmers* are among the first books in the series. These will be followed by *Java 2 Enterprise Edition*, *Java 2 Micro Edition*, *.NET A Technical Introduction*, *ASP .NET with Visual Basic .NET for Experienced Programmers*, *ASP .NET with C# for Experienced Programmers* and many more. Please visit `www.deitel.com` for continuous updates on all published and forthcoming *DEITEL™ Developer Series* titles.

The *DEITEL™ Developer Series* is divided into three subseries. The *A Technical Introduction* subseries provides IT managers and developers with detailed overviews of emerging technologies. The *A Programmer's Introduction* subseries is designed to teach the fundamentals of new languages and software technologies to programmers and novices from the ground up; these books discuss programming fundamentals, followed by brief introductions to more sophisticated topics. The *For Experienced Programmers* subseries is designed for seasoned developers seeking an intermediate-level treatment of new programming languages and technologies, without the encumbrance of introductory material; the books in this subseries move quickly to in-depth coverage of the features of the programming languages and software technologies being covered.

Tour of the Book

You are about to study one of today's most exciting and rapidly developing computer programming languages. Mastering Java will help you develop powerful business and personal computer-applications software. In this section, we take a tour of the many capabilities of Java you will study in *Java How to Program, Fifth Edition*.

Chapter 1—Introduction to Computers, the Internet and the Web—discusses what computers are, how they work and how they are programmed. The chapter gives a brief history of the development of programming languages from machine languages, to assembly languages, to high-level languages. The origin of the Java programming language is discussed. The chapter includes an introduction to a typical Java programming environment. The chapter also introduces object technology, the Unified Modeling Language and design patterns.

Chapter 2—Introduction to Java Applications—provides a lightweight introduction to programming *applications* in the Java programming language. The chapter introduces nonprogrammers to basic programming concepts and constructs. The programs in this chapter illustrate how to display data on the screen to the user and how to obtain data from the user at the keyboard. Some of the input and output is performed by using *graphical user interface (GUI)* components. Chapter 2 also provides detailed treatments of *decision making* and *arithmetic operations*.

Chapter 3—Introduction to Java Applets—introduces Java *applets*, which are Java programs designed to be transported over the Internet and executed in Web browsers (like Netscape Navigator and Microsoft Internet Explorer). The chapter shows several of the demonstration applets supplied with the J2SDK. We then write Java applets that perform tasks similar to the programs of Chapter 2, and we explain the similarities and differences between applets and applications.

Chapter 4—Control Statements: Part 1—focuses on the program-development process. The chapter discusses how to take a *problem statement* and from it develop a working Java program, including performing intermediate steps in *pseudocode*. The chapter introduces some primitive types and simple control statements for decision making (`if` and `if...else`) and repetition (`while`). We examine counter-controlled repetition and sentinel-controlled repetition, and introduce Java's increment, decrement and assignment operators. The chapter uses simple UML activity diagrams to show the flow of control through each of the control statements.

Chapter 5—Control Statements: Part 2—continues the discussions of Java control statements with examples of the `for` repetition statement, the `do...while` repetition statement, the `switch` selection statement, the `break` statement and the `continue` statement. The chapter also contains a discussion of Java logical operators.

Chapter 6—Methods—takes a deeper look inside objects. Objects contain data called fields and executable units called methods. We discuss class-library methods and build our own methods. For computer-science courses, the chapter also presents a discussion of recursion. The techniques presented in Chapter 6 are essential to the production of properly structured programs, especially the larger programs that system programmers and application programmers are likely to develop. The topic of method overloading (i.e., allowing multiple methods to have the same name as long as they have different "signatures") is motivated and explained clearly. We also introduce *events* and *event handling*.

Chapter 7—Arrays—explores processing lists and tables of values. Arrays in Java are objects, further evidence of Java's commitment to object orientation. We discuss the structuring of data into arrays of data items of the same type. The chapter presents numerous examples of both single-dimensional arrays and multidimensional arrays. Examples in the chapter investigate common array manipulations, printing histograms, passing arrays to methods and an introduction to the field of survey data analysis (with simple statistics). A feature of this chapter is the discussion of elementary sorting and searching techniques and the presentation of binary searching as a dramatic improvement over linear searching.

Chapter 8—Object-Based Programming—begins our deeper discussion of classes. The chapter represents a wonderful opportunity for teaching data abstraction the "right way"—through a language (Java) expressly devoted to implementing new types. The chapter focuses on the essence and terminology of classes and objects. The chapter discusses implementing Java classes, accessing class members, enforcing information hiding with access modifiers, separating interface from implementation, using access methods and utility methods and initializing objects with constructors. The chapter discusses declaring and using constants, *composition*, the `this` reference, `static` class members and examples of popular abstract data types such as stacks and queues. The chapter introduces the `package` statement and discusses how to create reusable packages.

Chapter 9—Object-Oriented Programming: Inheritance—introduces one of the most fundamental capabilities of object-oriented programming languages, inheritance,

which is a form of software reusability in which new classes are developed quickly and easily by absorbing the capabilities of existing classes and adding appropriate new capabilities. The chapter discusses the notions of superclasses and subclasses, access modifier `protected`, direct superclasses, indirect superclasses, use of constructors in superclasses and subclasses, and software engineering with inheritance. The chapter compares inheritance ("is a" relationships) with composition ("has a" relationships).

Chapter 10—Object-Oriented Programming: Polymorphism—deals with another fundamental capability of object-oriented programming, namely polymorphic behavior. This style of programming is typically used to implement today's popular GUI frameworks, such as Java's Swing. This chapter distinguishes between `abstract` classes and concrete classes, and introduces interfaces—Java's replacement for the dangerous (albeit powerful) feature of C++ called multiple inheritance. The chapter presents the powerful concept of *nested classes* that help hide implementation details. Then, the chapter demonstrates our first GUI-based applications as part of a more complete introduction to event handling. In this section, we use nested classes to define the event handlers for several GUI components. A feature of this chapter is its three polymorphism case studies—a payroll system, a shape hierarchy headed up by an `abstract` class and a shape hierarchy headed up by an interface.

Chapter 11—Strings and Characters—deals with processing words, sentences, characters and groups of characters. We present classes `String`, `StringBuffer`, `Character` and `StringTokenizer`. We also present Java's API for regular expressions (new to J2SE 1.4), which enables programs to search strings for sequences of characters that match specified patterns.

Chapter 12—Graphics and Java2D—is the first of several chapters that present Java's graphical and multimedia capabilities. We discuss graphics contexts and graphics objects; drawing strings, characters and bytes; color and font control; screen manipulation and paint modes and drawing lines, rectangles, rounded rectangles, three-dimensional rectangles, ovals, arcs and polygons. We introduce the Java2D API, which provides powerful graphics capabilities. Figure 12.22 is an example of how easy it is to use the Java2D API to create complex graphics effects such as textures and gradients.[6]

Chapter 13—Graphical User Interface Components: Part 1—introduces several of Java's *Swing components* for creating programs with user-friendly graphical user interfaces (GUIs). These *platform-independent* GUI components are written entirely in Java, providing them with great flexibility. Swing components can be customized to look like the computer platform on which the program executes, or they can use the standard Java look-and-feel to provide an identical user interface on all computer platforms. GUI development is a huge topic, so we divided it into two chapters. These chapters cover the material in sufficient depth to enable you to build rich user interfaces. The chapter illustrates GUI principles, the `javax.swing` hierarchy, labels, buttons, lists, textfields, combo boxes, checkboxes, radio buttons, panels, handling mouse events, handling keyboard events and layout managers to position components. The chapter enhances our discussions of event handling.

Chapter 14—Graphical User Interface Components: Part 2—continues the Swing discussion started in Chapter 13. Through its programs, tables and line drawings, the

---

6. Our companion book, *Advanced Java 2 Platform How to Program*, presents the Java 3D API for building three-dimensional worlds.

chapter illustrates GUI design principles, textareas, extending Swing components, sliders, windows, menus, pop-up menus, changing the look-and-feel, multiple-document interfaces, tabbed panes and using advanced layout managers.[7]

Chapter 15—Exception Handling—is one of the most important chapters in the book from the standpoint of building "mission-critical" or "business-critical" applications. Programmers need to be concerned with, "What happens when the component I call on to do a job experiences difficulty? How will that component signal that it had a problem?" To use a Java component, you need to know not only how that component behaves when "things go well," but also what *exceptions* that component *throws* when "things go poorly." The chapter distinguishes between rather serious system `Error`s and `Exception`s. The chapter discusses the vocabulary of exception handling, including *try blocks*, *catch clauses* and *finally clauses*. The chapter also introduces the new chained-exception facility in J2SE 1.4. The material in this chapter is crucial to many of the examples in the remainder of the book.

Chapter 16—Multithreading—deals with developing Java programs that can perform multiple activities concurrently. Computers used to be built with a single, rather expensive processor. Today, processors are becoming so inexpensive that it is possible to build computers with many processors that work in parallel—such computers are called *multiprocessors*. The trend is clearly towards computers that can perform many tasks in parallel. As we will see, multithreading is effective even on single-processor systems. This chapter presents multithreaded programs that demonstrate the problems that can occur in concurrent programming. A feature of the chapter is the extensive set of examples that show these problems and how to solve them. The chapter discusses threads and thread methods. It walks through the various thread states and state transitions with a graphical representation of a thread's life cycle. We discuss thread priorities and thread scheduling. We examine a producer/consumer relationship without synchronization, observe the problems that occur and solve the problem with thread synchronization. We implement a producer/consumer relationship with a circular buffer and proper synchronization with a monitor. We discuss daemon threads that "hang around" and perform tasks when processor time is available. We discuss interface `Runnable` which enables objects to run as threads without having to subclass class `Thread`.

Chapter 17—Files and Streams—deals with input/output that is accomplished through streams of data directed to and from files. In this chapter, we translate objects into a persistent format. Being able to store data in files or move it across networks (Chapter 18) makes it possible for programs to save data and to communicate with each other. The chapter begins with an introduction to the data hierarchy from bits, to bytes, to fields, to records, to files. Next, Java's simple view of files and streams is presented. We show how programs pass data to secondary storage devices, like disks, and how programs retrieve data already stored on those devices. We discuss class `File` which programs use to obtain information about files and directories. We explain how objects can be output to, and input from, secondary storage devices. We also introduce the high-performance, New I/O (NIO) APIs (introduced in J2SE 1.4).

Chapter 18—Networking—deals with Java programs that communicate over computer networks. This chapter presents Java's lowest level networking capabilities. The

---

7. Chapters 1–6 of *Advanced Java 2 Platform How to Program* present more advanced GUI concepts.

chapter examples illustrate an applet interacting with the browser in which it executes, creating a mini Web browser, communicating between two Java programs using streams-based sockets and communicating between two Java programs using packets of data. A key feature of the chapter is the implementation of a collaborative client/server Tic-Tac-Toe game in which two clients play Tic-Tac-Toe against each other arbitrated by a multi-threaded server—great stuff! The capstone example in the chapter is the Deitel Messenger case study, which simulates many of today's popular instant-messaging applications that enable computer users to communicate with friends, relatives and coworkers over the Internet. This 1,130-line, multithreaded, client/server case study uses most of the programming techniques presented up to this point in the book. The chapter also continues our discussion of J2SE 1.4's NIO APIs with an introduction to selectors and non-blocking I/O for implementing high-performance network servers.[8]

Chapter 19—Multimedia: Images, Animation and Audio—presents some of Java's capabilities for making programs come alive through multimedia. The chapter discusses images and image manipulation, audio and animation. We present a LIVE-CODE image-map application with the icons from the programming tips shown earlier in the preface and that appear throughout the book. As the user moves the mouse pointer across the icons, the tip type is displayed. Once you have read the chapter, you will be eager to try out all these techniques, so we have included many exercises to challenge and entertain you.

Chapter 20—Data Structures—is particularly valuable in second- and third-level university courses. The chapter discusses the techniques used to create and manipulate dynamic data structures, such as linked lists, stacks, queues and trees. For each type of data structure, we present examples with sample outputs. Although it is valuable to know how these classes are implemented, Java programmers will quickly discover that most of the data structures they need are available in class libraries, such as Java's own `java.util` that we discuss in Chapters 21–22.

Chapter 21—Java Utilities Package and Bit Manipulation—presents several `java.util` classes and discusses Java's bit-manipulation operators. One particularly useful class is `Vector`—a dynamic array that can grow and shrink as necessary. We also discuss `Stack`, `Hashtable`, `Properties`, `Random` and `BitSet`.

Chapter 22—Collections—discusses the `java.util` classes of the Collections API that provide predefined implementations of many of the data structures discussed in Chapter 20. Collections provide Java programmers with a standard set of data structures for storing and retrieving data and a standard set of algorithms (i.e., procedures) that allow programmers to manipulate the data (such as searching for particular data items and sorting data into ascending or descending order). The chapter examples demonstrate collections, such as linked lists, trees, maps and sets, and algorithms for searching, sorting, finding the maximum value, finding the minimum value and so on.

Chapter 23—Java Database Connectivity with JDBC™—discusses Java's support for databases. Today's most popular database systems are relational databases. We present examples using IBM's Cloudscape—a pure-Java database management system. This chapter introduces JDBC and uses it to connect to a Cloudscape database, then to manipu-

---

8. Our companion book, *Advanced Java 2 Platform How to Program*, offers a much deeper treatment of networking and distributed computing, with topics including remote method invocation (RMI), Java 2 Enterprise Edition, wireless Java (and the Java 2 Micro Edition) and Common Object Request Broker Architecture (CORBA).

late its content. We use the Structured Query Language (SQL) to extract information from, and insert information into, a database. The following chapters on servlets and JavaServer Pages use the techniques shown in this chapter to build data-driven Web applications.

Chapter 24—Servlets—discusses servlets, which typically extend the functionality of Web servers. Servlets are effective for developing Web-based solutions that interact with databases on behalf of clients, dynamically generate custom content to be displayed by browsers, and maintain unique session information for each client. The Java servlet API allows developers to add functionality to Web servers for handling client requests. Servlets also are reusable across Web servers and across platforms. This chapter demonstrates the Web's request/response mechanism (primarily with HTTP `get` and `post` requests), redirecting requests to other resources and interacting with databases through JDBC. The chapter features three-tier client/server application that tracks users' responses to a survey.

Chapter 25—JavaServer Pages (JSP)—introduces an extension of servlet technology called JavaServer Pages (JSP). JSPs enable delivery of dynamically generated Web content and are used primarily by Web designers and others who are not familiar with Java programming. JSPs may contain Java code in the form of scriptlets. To increase performance, each JSP is compiled into a Java servlet—this normally occurs the first time a JSP is requested by a client. Subsequent client requests are fulfilled by the compiled servlet. This chapter features a three-tier client/server guest-book application that stores guest information in a database.

Appendix A—Operator Precedence Chart—lists each of the Java operators and indicates their relative precedence and associativity.

Appendix B—ASCII Character Set—lists the characters of the ASCII (American Standard Code for Information Interchange) character set and indicates the character code value for each. Java uses the Unicode character set with 16-bit characters for representing all of the characters in the world's "commercially significant" languages. Unicode includes ASCII as a subset.

Appendix C—Number Systems—discusses the binary (base 2), decimal (base 10), octal (base 8) and hexadecimal (base 16) number systems. This material is valuable for introductory courses in computer science and computer engineering.

Appendices D–F contain the implementation of our case study on Object-Oriented Design with the UML. These are discussed in the overview of the case study.

Appendix G—Unicode—discusses the Unicode character set, which enables Java to display information in many languages. The appendix provides a sample Java program that displays "Welcome to Unicode" in several different languages.

(Optional) A Tour of the Case Study on Object-Oriented Design with the UML

In this and the next section, we tour the two optional major features of the book—the optional case study of object-oriented design with the UML and our introduction to design patterns. The case study involving object-oriented design with the UML is an important addition to *Java How to Program, Fifth Edition*. This tour previews the contents of the "Thinking About Objects" sections and discusses how they relate to the case study. After completing this case study, you will have completed an object-oriented design and implementation for a significant Java application.

Section 1.15—Thinking About Objects: Introduction to Object Technology and the Unified Modeling Language
This section introduces the object-oriented design case study with the UML. We provide a general background of what objects are and how they interact with other objects. We also discuss briefly the state of the software-engineering industry and how the UML has influenced object-oriented analysis and design processes.

Section 2.9—(Optional Case Study) Thinking About Objects: Examining the Problem Statement
Our case study begins with a *problem statement* that specifies the requirements for a system that we will create. In this case study, we design and implement a simulation of an elevator system in a two-story building. We provide the design of our elevator system after investigating the structure and behavior of object-oriented systems in general. We discuss how the UML will facilitate the design process in subsequent "Thinking About Objects" sections by providing us with several types of diagrams to model our system. Finally, we provide a list of URL and book references on object-oriented design with the UML. You might find these references helpful as you proceed through our case-study presentation.

Section 3.7—(Optional Case Study) Thinking About Objects: Identifying the Classes in the Problem Statement
In this section, we begin to design the elevator simulation. We identify the classes, or "building blocks," of our simulation by extracting the nouns and noun phrases from the problem statement. We arrange these classes into a UML class diagram that describes the class structure of our simulation. The class diagram also describes relationships, known as *associations*, among classes (for example, a person has an association with the elevator, because the person rides the elevator).

Section 4.14—(Optional Case Study) Thinking About Objects: Identifying Class Attributes
A class contains both *attributes* (data) and *operations* (behaviors). This section focuses on the attributes of the classes discussed in Section 3.7. As we see in later sections, changes in an object's attributes often affect the behavior of that object. To determine the attributes for the classes in our case study, we extract the adjectives describing the nouns and noun phrases (which defined our classes) from the problem statement, then place the attributes in the class diagram we create in Section 3.7.

Section 5.11—(Optional Case Study) Thinking About Objects: Identifying Objects' States and Activities
An object, at any given time, occupies a specific condition called a *state*. A *state transition* occurs when that object receives a message to change state. The UML provides the *state-chart diagram*, which identifies the set of possible states that an object may occupy and models that object's state transitions. An object also has an *activity*—the work performed by an object in its lifetime. The UML provides the *activity diagram*—a flowchart that models an object's *activity*. In this section, we use both types of diagrams to begin modeling specific behavioral aspects of our elevator simulation, such as how a person rides the elevator and how the elevator responds when a button is pressed on a given floor.

Section 6.15—(Optional Case Study) Thinking About Objects: Identifying Class Operations

In this section, we identify the operations, or services, of our classes. We extract from the problem statement the verbs and verb phrases that specify the operations for each class. We then modify the class diagram of Section 3.7 to include each operation with its associated class. At this point in the case study, we will have gathered all information possible from the problem statement. However, as future chapters introduce such topics as inheritance, event-handling and multithreading, we will modify our classes and diagrams.

Section 7.10—(Optional Case Study) Thinking About Objects: Collaboration Among Objects

At this point, we have created a "rough sketch" of the model for our elevator system. In this section, we see how it works. We investigate the behavior of the simulation by discussing *collaborations*—messages that objects send to each other to communicate. The class operations that we discovered in Section 6.15 turn out to be the collaborations among the objects in our system. We determine the collaborations in our system, then collect them into a *collaboration diagram*—the UML diagram for modeling collaborations. This diagram reveals which objects collaborate and when. We present a collaboration diagram of the people entering and exiting the elevator.

Section 8.17—(Optional Case Study) Thinking About Objects: Starting to Program the Classes for the Elevator Simulation

In this section, we take a break from designing the behavior of our system. We begin the implementation process to emphasize the material discussed in Chapter 8. Using the UML class diagram of Section 3.7 and the attributes and operations discussed in Sections 4.14 and 6.15, we show how to implement a class in Java from a design. We do not implement all classes—because we have not completed the design process. Working from our UML diagrams, we create code for the `Elevator` class.

Section 10.12—(Optional Case Study) Thinking About Objects: Incorporating Inheritance into the Elevator Simulation

Chapters 9–10 discuss object-oriented programming. We consider inheritance—classes sharing similar characteristics may inherit attributes and operations from a "base" class. In this section, we investigate how our elevator simulation can benefit from using inheritance. We document our discoveries in a class diagram that models inheritance relationships—the UML refers to these relationships as *generalizations*. We modify the class diagram of Section 3.7 by using inheritance to group classes with similar characteristics.

Section 11.9—(Optional Case Study) Thinking About Objects: Event Handling

In this section, we include interfaces necessary for the objects in our elevator simulation to send messages to other objects. In Java, objects often communicate by sending an *event*—a notification that some action has occurred. The object receiving the event then performs an action in response to the type of event received—this is known as *event handling*. In Section 7.10, we outlined the message passing, or the collaborations, in our model, using a collaboration diagram. We now modify this diagram to include event handling, and, as an example, we explain in detail how doors in our simulation open upon the elevator's arrival.

Section 12.9—(Optional Case Study) Thinking About Objects: Designing Interfaces with the UML

In this section, we design a class diagram that models the relationships between classes and interfaces in our simulation—the UML refers to these relationships as *realizations*. In addition, we list all operations that each interface provides to the classes. Lastly, we show how to create the Java classes that implement these interfaces.

Section 13.17 - (Optional Case Study) Thinking About Objects: Use Cases

Chapter 13 discusses user interfaces that enable a user to interact with a program. In this section, we discuss the interaction between our elevator simulation and its user. Specifically, we investigate the scenarios that may occur between the application user and the simulation itself—this set of scenarios is called a *use case*. We model these interactions, using *use-case diagrams* of the UML.

Section 14.13—(Optional Case Study) Thinking About Objects: Model-View-Controller

We designed our system to consist of three components, each having a distinct responsibility. By this point in the case study, we have almost completed the first component, called the *model*, which contains data that represent the simulation. We design the *view*—the second component, dealing with how the model is displayed—in Section 19.7. We design the *controller*—the component that allows the user to control the model—in this section. A system such as ours that uses the model, view and controller components is said to adhere to *Model-View-Controller* (*MVC*) architecture. In this section, we explain the advantages of using this architecture to design software. We use the UML *component diagram* to model the three components, then implement this diagram as Java code.

Section 16.11—(Optional Case Study) Thinking About Objects: Multithreading

In this section, we declare certain objects as "threads" to enable these objects to operate concurrently. We modify the collaboration diagram originally presented in Section 7.10 (and modified in Section 11.9) to incorporate multithreading. We present the UML *sequence diagram* for modeling interactions in a system. This diagram emphasizes the chronological ordering of messages. We use a sequence diagram to model how a person inside the simulation interacts with the elevator. This section concludes the design of the model portion of our simulation. We design how this model is displayed in Section 19.7, then implement this model as Java code in Appendix E.

Section 19.7—(Optional Case Study) Thinking About Objects: Animation and Sound in the View

This section designs the view, which specifies how the model portion of the simulation is displayed. Chapter 19 presents several techniques for integrating sound and animation in programs. This section uses some of these techniques to incorporate sound and animation into our elevator simulation. Specifically, this section deals with animating the movements of people and our elevator, generating sound effects and playing "elevator music" when a person rides the elevator. This section concludes the design of our elevator simulation. Appendices D, E and F implement this design as a 3,320-line, fully operational Java program.

Appendix D—Elevator Events and Listener Interfaces

[Note: This appendix is on the CD that accompanies this book.] As we discussed in Section 11.9, several objects in our simulation interact with each other by sending messag-

es, called events, to other objects wishing to receive these events. The objects receiving the events are called *listener objects*—these must implement *listener interfaces*. In this appendix, we implement all event classes and listener interfaces used by the objects in our simulation.

Appendix E—Elevator Model

[Note: This appendix is on the CD that accompanies this book.] The majority of the case study involved designing the model (i.e., the data and logic) of the elevator simulation. In this appendix, we implement that model in Java. Using all the UML diagrams we created, we present the Java classes necessary to implement the model. We apply the concepts of object-oriented design with the UML and object-oriented programming and Java that you learned in the chapters.

Appendix F—Elevator View

[Note: This appendix is on the CD that accompanies this book.] This final appendix implements our display of the elevator simulation. We use the same approach to implement the view as we used to implement the model—we create all the classes required to run the view, using the UML diagrams and key concepts discussed in the chapters. By the end of this appendix, you will have completed an "industrial-strength" design and implementation of a large-scale system. You should feel confident tackling larger systems, such as the 10,000-line Enterprise Java case study we present in our companion book *Advanced Java 2 Platform How to Program* and the kinds of applications that professional software engineers build. Hopefully, you will move on to even deeper study of object-oriented design with the UML.

(Optional) A Tour of the "Discovering Design Patterns" Sections

Our treatment of design patterns is spread over five optional sections of the book. We overview those sections here.

Section 10.12—(Optional) Discovering Design Patterns: Introducing Creational, Structural and Behavioral Design Patterns

This section lists the sections in which we discuss the various design patterns. We divide the discussion of each section into creational, structural and behavioral design patterns. Creational patterns provide ways to instantiate objects, structural patterns deal with organizing objects and behavioral patterns deal with interactions between objects. The remainder of the section introduces some of these design patterns, such as the Singleton, Proxy, Memento and State design patterns. Finally, we provide several URLs for further study on design patterns.

Section 14.14—(Optional) Discovering Design Patterns: Design Patterns Used in Packages `java.awt` and `javax.swing`

This section contains most of our design-patterns discussion. Using the material on Java Swing GUI components in Chapters 13 and 14, we investigate some examples of pattern use in packages `java.awt` and `javax.swing`. We discuss how these classes use the Factory Method, Adapter, Bridge, Composite, Chain-of-Responsibility, Command, Observer, Strategy and Template Method design patterns. We motivate each pattern and present examples of how to apply them.

Section 16.12—(Optional) Discovering Design Patterns: Concurrent Design Patterns
Developers have discovered several design patterns since those described by the gang of four. In this section, we discuss concurrency design patterns, including Single-Threaded Execution, Guarded Suspension, Balking, Read/Write Lock and Two-Phase Termination—these solve various design problems in multithreaded systems. We investigate how class `java.lang.Thread` uses concurrency patterns.

Section 18.12—(Optional) Discovering Design Patterns: Design Patterns Used in Packages `java.io` and `java.net`
Using the material on files, streams and networking in Chapters 17 and 18, we investigate some examples of pattern use in packages `java.io` and `java.net`. We discuss how these classes use the Abstract Factory, Decorator and Facade design patterns. We also consider architectural patterns, which specify a set of subsystems—aggregates of objects that each collectively comprise a major system responsibility—and how these subsystems interact with each other. We discuss the popular Model-View-Controller and Layers architectural patterns.

Section 22.12—(Optional) Discovering Design Patterns: Design Patterns Used in Package `java.util`
Using the material on data structures and collections in Chapters 20–22, we investigate pattern use in package `java.util`. We discuss how these classes use the Prototype and Iterator design patterns. This section concludes the discussion on design patterns. After finishing the *Discovering Design Patterns* material, you should be able to recognize and use key design patterns and have a better understanding of the workings of the Java API. After completing this material, we recommend that you move on to the Gang-of-Four book.

Acknowledgments

One of the great pleasures of writing a textbook is acknowledging the efforts of the many people whose names may not appear on the cover, but whose hard work, cooperation, friendship, and understanding were crucial to the production of the book.

Several people at Deitel & Associates, Inc. devoted long hours to this project. We would like to acknowledge the efforts of our full-time Deitel & Associates, Inc. colleagues Tem Nieto, Sean Santry, Su Zhang and Jeff Listfield.

- Tem Nieto is a graduate of the Massachusetts Institute of Technology. Tem teaches XML, Java, Internet and Web, C, C++ and Visual Basic seminars and works with us on textbook writing, course development and multimedia authoring efforts. He is co-author with us of several books, including *Internet & World Wide Web How to Program (Second Edition), XML How to Program, Visual Basic .NET How to Program* and *C# How to Program*. In *Java How to Program, Fifth Edition* Tem co-authored Chapters 12–14 and 22 and the Special Section entitled "Building Your Own Compiler" in Chapter 20.

- Sean Santry, a graduate of Boston College (Computer Science and Philosophy) and co-author of *Advanced Java 2 Platform How to Program* and *Java Web Services for Experienced Programmers*, edited the entire manuscript, designed and implemented the Deitel Messenger networking application in Chapter 18 (Networking), contributed to the design and updated the optional case study on OOD/UML and updated the optional design patterns introduction.

- Su Zhang holds B.Sc and a M.Sc degrees in Computer Science from McGill University. Her graduate research included modeling and simulation, real-time systems and Java technology. She is co-author with us on *Java Web Services for Experienced Programmers* and has contributed to other Deitel publications, including *Advanced Java 2 Platform How to Program* and *Python How to Program*. Su helped update several chapters and created the examples that introduce features new to Java 1.4, such as the New I/O APIs (covered in Chapters 17 and 18) and chained exceptions (covered in Chapter 15).

- Jeff Listfield is a Computer Science graduate of Harvard College. His course work included classes in computer graphics, networks and computational theory and he has programming experience in C, C++, C#, Java, Perl and Lisp. Jeff is our co-author on *C# How to Program*, *C# A Programmer's Introduction* and *C# for Experienced Programmers*, and contributed to *Perl How to Program*. Jeff helped update several chapters in the book, and wrote new examples and sections on regular expressions in Chapter 11, sorting in Chapter 22 and parts of the New I/O API sections in Chapters 17 and 18).

We are fortunate to have been able to work on this project with the talented and dedicated team of publishing professionals at Prentice Hall. We especially appreciate the extraordinary efforts of our computer science editor, Petra Recter and her boss—our mentor in publishing—Marcia Horton, Editorial Director of Prentice-Hall's Engineering and Computer Science team. Tom Manshreck did a marvelous job as production manager. Jennifer Cappello did a wonderful job managing the review process.

The *Java 2 Multimedia Cyber Classroom, Fifth Edition* was developed in parallel with *Java How to Program, Fifth Edition*. We sincerely appreciate the "new media" insight, savvy and technical expertise of our e-media editor-in-chief, mentor and friend Mark Taub. He and our e-media editor, Karen Mclean, did a remarkable job bringing the *Java 2 Multimedia Cyber Classroom, Fifth Edition* to publication under a tight schedule.

We owe special thanks to Tamara Newnam (`smart_art@earthlink.net`) who did the art work for our programming tips icons and the cover. She created the delightful bug creature who shares with you the book's programming tips.

We sincerely appreciate the efforts of our 70 fourth edition post-publication reviewers and our fifth edition reviewers:

Sun Microsystems Reviewers
Dibyendu Baksi (Sun Microsystems)
Konstantin Kladko (Sun Microsystems)
Doug Kohlert (Sun Microsystems)
Peter Jones (Sun Microsystems)
Paul Monday (Sun Microsystems)
Tomas Pavek (Sun Microsystems)
Brandon Taylor (Sun Microsystems)

Academic Reviewers
Rekha Bhowmik (St. Cloud State University)
Clint Bickmore (Front Range Community College)
Brian Blake (Georgetown University)
Ayad Boudiab (Georgia Perimiter College)

Chadi Boudiab (Georgia Perimiter College)
Michael Buckley (State University of New York-Buffalo)
James Chegwidden (Tarrant County College)
Deborah Coleman (Rochester Institute of Technology)
Don Francis Costello (University of Nebraska)
Balazs Csizmazia (University of Klagenfurt)
Tamara Dinev (Florida Atlantic University)
Sarah Fix (The Career Center High School)
Bill Freitas (The Lawrenceville School)
Thomas Graffte (Strayer University)
Balaji Janamanchi (Texas Tech University)
Charles Lake (Faulkner State Community College)
Brian Larson (Modesto Junior College)
Hong Lin (DeVry University)
David McKain (Lakota East High School)
Andy Novobilski (University of Tennessee-Chattanooga)
Richard Ord (University of California, San Diego)
Gavin Osborne (Saskatchewan Institute of Applied Sci.& Tech.)
Merrill Parker (Chattanooga State Technical Community College)
Donna Reese (Mississippi State University)
Craig Slinkman (University of Texas, Arlington)
Gidget Smith (Arkansas State University)
Ron Sones (James Madison University)
Mahendran Velauthapillai (Georgetown University)
Loran Walker (Lawrence Technological University)
Warren Wiltsie (Fairleigh Dickinson University)

Other Industry Reviewers
Shishir Abhyanker (Accenture)
Sinan Alhir (Independent Consultant)
Richard Bonneau (IONA Technologies)
Columbus Brown (IBM)
Carl Burnham (Southpoint)
Brian Cook (Zurich Insurance)
Jonathan Earl (Independent Consultant)
Ron Felice (Omniware Development)
Karl Frank (TogetherSoft Corporation)
Kyle Gabhart (Independent Consultant)
Johan Galle (E2S)
Mark Grand (ClickBlocks, LLC)
Ajay Gupta (American Airlines, Inc.)
Kevlin Henney (Curbralan, Ltd.)
Ethan Henry (Sitraka)
Anne Horton (AT&T Laboratories)
James Huddleston (Independent Consultant)
Terrell Hull (Sun Certified Java Architect, Rational Qualified Practitioner)
Sachin Korgaonkar (Idealake Technologies, Pvt. Ltd.)

Don Kostuch (You Can C Clearly Now)
Krishna Kunchithapadam (Oracle Corporation)
Paul McLachlan (Compuware Corporation)
Davyd Norris (Rational)
Bill O'Farrell (IBM)
Praveen Sadhu (Infodat Solutions, Inc.)
Cameron Skinner (Embarcadero Technologies/OMG)
Stephen Tockey (Construx Software/OMG)
Kim Topley (Keyboard Edge, Ltd.)
Sudhir Upadhyay (BEA Systems, Inc.)
John Varghese (UBS Warburg)
Bing Xue (Siemens)
Hadar Ziv (eBuilt, Inc.)

Under a tight time schedule, they scrutinized every aspect of the text and made countless suggestions for improving the accuracy and completeness of the presentation.

Well, there you have it! We have worked hard to create this book and its optional Cyber Classroom version. The book is loaded with LIVE-CODE™ examples, programming tips, self-review exercises and answers, challenging exercises and projects, and numerous study aids to help you master the material. Java is a powerful programming language that will help you write programs quickly and effectively. And Java is a language that scales nicely into the realm of enterprise-systems development to help organizations build their key information systems. As you read the book, we would sincerely appreciate your comments, criticisms, corrections and suggestions for improving the text. Please address all correspondence to:

```
deitel@deitel.com
```

We will respond promptly, and we will post corrections and clarifications on our Web site,

```
www.deitel.com
```

We hope you enjoy learning with *Java How to Program, Fifth Edition* as much as we enjoyed writing it!

> *Dr. Harvey M. Deitel*
> *Paul J. Deitel*

About the Authors

Dr. Harvey M. Deitel, Chairman of Deitel & Associates, Inc., has 40 years experience in the computing field including extensive industry and academic experience. He is one of the world's leading computer science instructors and seminar presenters. Dr. Deitel earned B.S. and M.S. degrees from the Massachusetts Institute of Technology and a Ph.D. from Boston University. He has 20 years of college teaching experience including earning tenure and serving as the Chairman of the Computer Science Department at Boston College before founding Deitel & Associates, Inc. with his son Paul J. Deitel. He is author or co-author of several dozen books and multimedia packages. With translations published in Japanese,

Russian, Spanish, Italian, Basic Chinese, Traditional Chinese, Korean, French, Polish, Turkish, Urdu, Greek, and Portuguese, Dr. Deitel's texts have earned international recognition. Dr. Deitel has delivered professional seminars internationally to major corporations, government organizations and various branches of the military.

Paul J. Deitel, CEO of Deitel & Associates, Inc., is a graduate of the Massachusetts Institute of Technology's Sloan School of Management where he studied Information Technology. Through Deitel & Associates, Inc. he has delivered programming-language seminars to major corporations, government organizations and various branches of the military. He has lectured on Java and C++ for the Boston Chapter of the Association for Computing Machinery, and has taught satellite-based courses through a cooperative venture of Deitel & Associates, Inc., Prentice Hall and the Technology Education Network. He and his father, Dr. Harvey M. Deitel, are the world's best-selling Computer Science textbook authors.

About Deitel & Associates, Inc.

Deitel & Associates, Inc. is an internationally recognized corporate training and content-creation organization specializing in Internet/World Wide Web software technology, e-business/e-commerce software technology and computer programming languages education. The company provides courses on Internet and World Wide Web programming, object technology and major programming languages. The founders of Deitel & Associates, Inc. are Dr. Harvey M. Deitel and Paul J. Deitel. The company's clients include many of the world's largest computer companies, government agencies, branches of the military and business organizations. Through its publishing partnership with Prentice Hall, Deitel & Associates, Inc. publishes leading-edge programming textbooks, professional books, interactive CD-ROM-based multimedia *Cyber Classrooms*, satellite courses and Web-based training courses. Deitel & Associates, Inc. and the authors can be reached via e-mail at

    deitel@deitel.com

To learn more about Deitel & Associates, Inc., its publications and its worldwide corporate on-site curriculum, see the last few pages of this book, visit:

    www.deitel.com

and subscribe to the free *DEITEL BUZZ ONLINE* e-mail newsletter at

    www.deitel.com/newsletter/subscribe.html

Individuals wishing to purchase Deitel books, Cyber Classrooms, Complete Training Courses and Web-based training courses can do so through

    www.deitel.com/books/index.html

Bulk orders by corporations and academic institutions should be placed directly with Prentice Hall. See the last few pages of this book for worldwide ordering details.