Welcome to Java and *Java SE 8 for Programmers, Third Edition*! This book presents leading-edge computing technologies for software developers.

We focus on software engineering best practices. At the heart of the book is the Deitel signature “live-code approach”—rather than using code snippets, we present concepts in the context of complete working programs that run on recent versions of Windows®, Linux®, and OS X®. Each complete code example is accompanied by live sample executions. All the source code is available at

http://www.deitel.com/books/javafp3/

**Keeping in Touch with the Authors**

As you read the book, if you have questions, send an e-mail to us at

deitel@deitel.com

and we’ll respond promptly. For updates on this book, visit

http://www.deitel.com/books/jfp3

subscribe to the *Deitel® Buzz Online* newsletter at

http://www.deitel.com/newsletter/subscribe.html

and join the Deitel social networking communities on

- Facebook® (http://www.deitel.com/deitelfan)
- Twitter® (@deitel)
- Google+™ (http://google.com/+DeitelFan)
- YouTube® (http://youtube.com/DeitelTV)
- LinkedIn® (http://linkedin.com/company/deitel-&-associates)

**Modular Organization**

*Java SE 8 for Programmers, 3/e,* is appropriate for programmers with a background in high-level language programming. It features a modular organization:

**Introduction**

- Chapter 1, Introduction to Java and Test-Driving a Java Application
- Chapter 2, Introduction to Java Applications; Input/Output and Operators
- Chapter 3, Introduction to Classes, Objects, Methods and Strings
xxvi Preface

Additional Programming Fundamentals
- Chapter 4, Control Statements: Part 1; Assignment, ++ and -- Operators
- Chapter 5, Control Statements: Part 2; Logical Operators
- Chapter 6, Methods: A Deeper Look
- Chapter 7, Arrays and ArrayLists
- Chapter 14, Strings, Characters and Regular Expressions
- Chapter 15, Files, Streams and Object Serialization

Object-Oriented Programming
- Chapter 8, Classes and Objects: A Deeper Look
- Chapter 9, Object-Oriented Programming: Inheritance
- Chapter 10, Object-Oriented Programming: Polymorphism and Interfaces
- Chapter 11, Exception Handling: A Deeper Look

Swing and JavaFX Graphical User Interfaces: Java 2D Graphics
- Chapter 12, Swing GUI Components: Part 1
- Chapter 13, Graphics and Java 2D
- Chapter 19, Swing GUI Components: Part 2
- Chapter 22, JavaFX GUI

Generic Collections, Lambdas and Streams
- Chapter 16, Generic Collections
- Chapter 17, Java SE 8 Lambdas and Streams
- Chapter 18, Generic Classes and Methods

Concurrency/Database
- Chapter 20, Concurrency
- Chapter 21, Accessing Databases with JDBC

Object-Oriented Design
- Chapter 23, ATM Case Study, Part 1: Object-Oriented Design with the UML
- Chapter 24, ATM Case Study Part 2: Implementing an Object-Oriented Design

New and Updated Features
Here are the updates we’ve made for Java SE 8 for Programmers, 3/e:
- Easy to use with Java SE 7 or Java SE 8. This book was published coincident with the release of Java SE 8. To meet the needs of our diverse audiences, we designed the book for professionals interested in Java SE 7, Java SE 8 or a mixture
of both. The Java SE 8 features (Fig. 4.1) are covered in Chapter 17 and in easy-to-include-or-omit sections book wide.

**Java SE 8 features**

- Lambda expressions
- Type-inference improvements
- `@FunctionalInterface` annotation
- Parallel array sorting
- Bulk data operations for Java Collections—filter, map and reduce
- Library enhancements to support lambdas (e.g., `java.util.stream`, `java.util.function`)
- Date & Time API (`java.time`)
- Java concurrency API improvements
- Static and default methods in interfaces
- Functional interfaces—interfaces that define only one abstract method and can include static and default methods
- JavaFX enhancements

**Fig. 4.1** | Java SE 8 features we discuss.

- **Java SE 8 lambdas, streams, and interfaces with default and static methods.**
  The most significant new features in Java SE 8 are lambdas and complementary technologies. In Chapter 17, you'll see that functional programming with lambdas and streams can help you write programs faster, more concisely, more simply, with fewer bugs and that are easier to parallelize (to get performance improvements on multi-core systems) than programs written with previous techniques (Fig. 4.2). You’ll see that functional programming complements object-oriented programming.

**Pre-Java-SE-8 topics**

<table>
<thead>
<tr>
<th>Pre-Java-SE-8 topics</th>
<th>Corresponding Java SE 8 discussions and examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 7, Arrays and ArrayLists</td>
<td>Sections 17.3–17.4 introduce basic lambda and streams capabilities that process one-dimensional arrays.</td>
</tr>
<tr>
<td>Chapter 10, Object-Oriented Programming: Polymorphism and Interfaces</td>
<td>Section 10.10 introduces the new Java SE 8 interface features (default methods, static methods and the concept of functional interfaces) that support functional programming with lambdas and streams.</td>
</tr>
<tr>
<td>Chapters 12 and 19, Swing GUI Components: Parts 1 and 2</td>
<td>Section 17.9 shows how to use a lambda to implement a Swing event-listener functional interface.</td>
</tr>
<tr>
<td>Chapter 14, Strings, Characters and Regular Expressions</td>
<td>Section 17.5 shows how to use lambdas and streams to process collections of String objects.</td>
</tr>
</tbody>
</table>

**Fig. 4.2** | Java SE 8 lambdas and streams discussions and examples. (Part 1 of 2.)
Java SE 7’s `try-with-resources` statement and the `AutoClosable` interface. `AutoClosable` objects reduce the likelihood of resource leaks when you use them with the `try-with-resources` statement, which automatically closes the `AutoClosable` objects. In this edition, we use `try-with-resources` and `AutoClosable` objects as appropriate starting in Chapter 15, Files, Streams, and Object Serialization.

**Java security.** We audited our book against the CERT Oracle Secure Coding Standard for Java: See this Preface’s Secure Java Programming section for more about CERT.

**Java NIO API.** We updated the file-processing examples in Chapter 15 to use features from the Java NIO (new IO) API.

**Java Documentation.** Throughout the book, we provide links to Java documentation where you can learn more about various topics that we present. For Java SE 7 documentation, the links begin with

http://docs.oracle.com/javase/7/

and for Java SE 8 documentation, the links begin with

http://download.java.net/jdk8/

These links could change when Oracle releases Java SE 8—possibly to links beginning with

http://docs.oracle.com/javase/8/

For any links that change after publication, we’ll post updates at

http://www.deitel.com/books/jfp3

**Swing and JavaFX GUI; Java 2D Graphics**

- **Swing GUI and Java 2D graphics.** Java’s Swing GUI is discussed in Chapters 12 and 19. Swing is now in maintenance mode—Oracle has stopped development...
New and Updated Features

and will provide only bug fixes going forward, however it will remain part of Java and is still widely used. Most of GUI-based legacy code in industry uses Swing GUI. Chapter 13 discusses Java 2D graphics.

- **JavaFX GUI.** Java’s GUI, graphics and multimedia technology going forward is JavaFX. In Chapter 22, we use JavaFX 2.2 with Java SE 7. We use Scene Builder—a drag-and-drop tool for creating JavaFX GUIs quickly and conveniently. It’s a standalone tool that you can use separately or with Java IDEs.

**Concurrency**

- **Concurrency for optimal multi-core performance.** In this edition, we were privileged to have as a reviewer Brian Goetz, co-author of *Java Concurrency in Practice* (Addison-Wesley). We updated Chapter 20, Concurrency, with Java SE 8 technology and idiom. We added a parallelSort vs. sort example that uses the Java SE 8 Date/Time API to time each operation and demonstrate parallelSort’s better performance on a multi-core system. We include a Java SE 8 parallel vs. sequential stream processing example, again using the Date/Time API to show performance improvements. Finally, we added a Java SE 8 CompletableFuture example that compares the relative performance of sequential and parallel execution of long-running calculations.

- **SwingWorker class.** We use class SwingWorker to create multithreaded user interfaces.

- **Concurrency is challenging.** There’s a great variety of concurrency features. We point out the ones that most developers should use and mention those that should be left to the experts.

**Getting Monetary Amounts Right**

- **Monetary amounts.** In the early chapters, for convenience, we use type double to represent monetary amounts. Due to the potential for incorrect monetary calculations with type double, class BigDecimal (which is a bit more complex) should be used to represent monetary amounts. We demonstrate BigDecimal in Chapters 8 and 22.

**Object Technology**

- **Object-oriented programming.** We use an early objects approach, reviewing the basic concepts and terminology of object technology in Chapter 1. Readers develop their first customized classes and objects in Chapter 3.

- **Early objects real-world case studies.** The early classes and objects presentation features Account, Student, AutoPolicy, Time, Employee, GradeBook and Card shuffling-and-dealing case studies, gradually introducing deeper OO concepts.

- **Inheritance, Interfaces, Polymorphism and Composition.** We use a series of real-world case studies to illustrate each of these OO concepts and explain situations in which each is preferred in building industrial-strength applications. We discuss Java SE 8’s improvements to the interface concept.
Preface

- **Exception handling.** We integrate basic exception handling early in the book then present a deeper treatment in Chapter 11. Exception handling is important for building “mission-critical” and “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.”

- **Class Arrays and ArrayList.** Chapter 7 covers class Arrays—which contains methods for performing common array manipulations—and class ArrayList—which implements a dynamically resizable array-like data structure. This follows our philosophy of getting lots of practice using existing classes while learning how to define your own classes.

- **Case Study: Developing an Object-Oriented Design and Java Implementation of an ATM.** Chapters 23–24 include a case study on object-oriented design with the UML (Unified Modeling Language™)—the industry-standard graphical language for modeling object-oriented systems. We design and implement the software for a simple automated teller machine (ATM). We analyze a typical requirements document that specifies the system to be built. We determine the classes needed to implement that system, the attributes the classes need to have, the behaviors the classes need to exhibit and specify how the classes must interact with one another to meet the system requirements. From the design we produce a completely coded Java implementation. Participants in our professional Java courses often report having a “light-bulb moment”—the case study helps them “tie it all together” and really understand Java-based object-oriented programming.

**Generic Collections**

- **Generic collections presentation.** We begin with generic class ArrayList in Chapter 7. Chapters 16–18 provide a deeper treatment of generic collections—showing how to use the built-in collections of the Java API. We show how to implement generic methods and classes. Lambdas and streams (introduced in Chapter 17) are especially useful for working with generic collections.

**Database**

- **JDBC.** Chapter 21 covers JDBC and uses the Java DB database management system. The chapter introduces Structured Query Language (SQL) and features an OO case study on developing a database-driven address book that demonstrates prepared statements.

**Secure Java Programming**

It’s difficult to build industrial-strength systems that stand up to attacks from viruses, worms, and other forms of “malware.” Today, via the Internet, such attacks can be instantaneous and global in scope. Building security into software from the beginning of the development cycle can greatly reduce vulnerabilities. We incorporate various secure Java coding practices into our discussions and code examples.
Teaching Approach

The CERT® Coordination Center (www.cert.org) was created to analyze and respond promptly to attacks. CERT—the Computer Emergency Response Team—is a government-funded organization within the Carnegie Mellon University Software Engineering Institute™. CERT publishes and promotes secure coding standards for various popular programming languages to help software developers implement industrial-strength systems that avoid the programming practices which leave systems open to attack.

We’d like to thank Robert C. Seacord, Secure Coding Manager at CERT and an adjunct professor in the Carnegie Mellon University School of Computer Science. Mr. Seacord was a technical reviewer for our book, *C11 for Programmers*, where he scrutinized our C programs from a security standpoint, recommending that we adhere to the *CERT C Secure Coding Standard*. This experience influenced our coding practices in *C++11 for Programmers* and *Java SE 8 for Programmers, 3/e* as well.

Teaching Approach

*Java SE 8 for Programmers, 3/e*, contains hundreds of complete working examples. We stress program clarity and concentrate on building well-engineered software.

Syntax Coloring. For readability, we syntax color the code, similar to the way most integrated-development environments and code editors syntax color the code. Our syntax-coloring conventions are:

- comments appear like this
- keywords appear like this
- constants and literal values appear like this
- errors appear like this
- all other code appears in black

Code Highlighting. We place yellow rectangles around each program’s key code.

Using Fonts for Emphasis. We place the key terms and the index’s page reference for each defining occurrence in bold text for easier reference. On-screen components are emphasized in the bold Helvetica font (e.g., the File menu) and Java program text in the Lucida font (e.g., `int x = 5;`).

Web Access. All of the source-code examples can be downloaded from:

- www.deitel.com/books/javafp3
- www.pearsonhighered.com/deitel

Objectives. The opening quotations are followed by a list of chapter objectives.

Illustrations/Figures. Abundant tables, line drawings, UML diagrams, programs and program outputs are included.

Programming Tips. We include programming tips to help you focus on important aspects of program development. These tips and practices represent the best we’ve gleaned from a combined seven decades of programming and teaching experience.

Good Programming Practice

*The Good Programming Practices call attention to techniques that will help you produce programs that are clearer, more understandable and more maintainable.*
Common Programming Error
Pointing out these Common Programming Errors reduces the likelihood that you’ll make them.

Error-Prevention Tip
These tips contain suggestions for exposing and removing bugs from your programs; many of the tips describe aspects of Java that prevent bugs from getting into programs.

Performance Tip 4.1
These tips highlight opportunities for making your programs run faster or minimizing the amount of memory that they occupy.

Portability Tip
The Portability Tips help you write code that will run on a variety of platforms.

Software Engineering Observation
The Software Engineering Observations highlight architectural and design issues that affect the construction of software systems, especially large-scale systems.

Look-and-Feel Observation
The Look-and-Feel Observations highlight graphical-user-interface conventions. These observations help you design attractive, user-friendly graphical user interfaces that conform to industry norms.

Index. We’ve included an extensive index. Defining occurrences of key terms are highlighted with a **bold** page number.

Software Used in *Java SE 8 for Programmers, 3/e*
All the software you’ll need for this book is available free for download from the Internet. See the Before You Begin section that follows this Preface for links to each download.

We wrote most of the examples in *Java SE 8 for Programmers, 3/e*, using the free Java Standard Edition Development Kit (JDK) 7. For the Java SE 8 modules, we used the OpenJDK’s early access version of JDK 8. In Chapter 22, we also used the Netbeans IDE. See the Before You Begin section that follows this Preface for more information.

Our *Java Fundamentals: Parts I, II, III and IV LiveLessons, 2/e* (summer 2014), video training product shows you what you need to know to start building robust, powerful software with Java. It includes 30+ hours of expert training synchronized with *Java SE 8 for Programmers, Third Edition*. Visit

http://www.deitel.com/livelessons
for information on purchasing Deitel LiveLessons video products online from Informit and Udemy. You may also access our LiveLessons videos if you have a subscription to Safari Books Online (http://www.safaribooksonline.com).

Acknowledgments

We’d like to thank Abbey Deitel and Barbara Deitel of Deitel & Associates, Inc. for long hours devoted to this project. Abbey co-authored Chapter 1 and this Preface, and she and Barbara painstakingly researched the new capabilities of Java SE 8.

We’re fortunate to have worked on this project with the dedicated publishing professionals at Prentice Hall/Pearson. We appreciate the extraordinary efforts and 19-year mentorship of our friend and professional colleague Mark L. Taub, Editor-in-Chief of Pearson Technology Group. Carole Snyder recruited distinguished members of the Java community to review the manuscript and managed the review process. Chuti Prasertsith designed the cover. John Fuller managed the book’s publication.

Reviewers

We wish to acknowledge the efforts of our recent editions reviewers—a distinguished group of Oracle Java team members, Oracle Java Champions, other industry professionals and academics. They scrutinized the text and the programs and provided countless suggestions for improving the presentation.

**Third Edition reviewers:** Lance Andersen (Oracle Corporation), Dr. Danny Coward (Oracle Corporation), Brian Goetz (Oracle Corporation), Evan Golub (University of Maryland), Dr. Huwei Guan (Professor, Department of Computer & Information Science, North Shore Community College), Manfred Riem (Java Champion), Simon Ritter (Oracle Corporation), Robert C. Seacord (CERT, Software Engineering Institute, Carnegie Mellon University), Khallai Taylor (Assistant Professor, Triton College and Adjunct Professor, Lonestar College—Kingwood), Jorge Vargas (Yumbling and a Java Champion), Johan Vos (LodgON and Oracle Java Champion) and James L. Weaver (Oracle Corporation and author of *Pro JavaFX 2*).

**Other recent editions reviewers:** Soundararajan Angusamy (Sun Microsystems), Joseph Bowbeer (Consultant), William E. Duncan (Louisiana State University), Diana Franklin (University of California, Santa Barbara), Edward F. Gehringer (North Carolina State University), Ric Heishman (George Mason University), Dr. Heinz Kabutz (JavaSpecialists.eu), Patty Kraft (San Diego State University), Lawrence Premkumar (Sun Microsystems), Tim Margush (University of Akron), Sue McFarland Metzger (Villanova University), Shyamal Mitra (The University of Texas at Austin), Peter Pilgrim (Consultant), Manjeet Rege, Ph.D. (Rochester Institute of Technology), Susan Rodger (Duke University), Amr Sabry (Indiana University), José Antonio González Seco (Parliament of Andalucía), Sang Shin (Sun Microsystems), S. Sivakumar (Astra Infotech Private Limited), Raghavan “Rags” Srinivas (Intuit), Monica Sweat (Georgia Tech), Vinod Varma (Astra Infotech Private Limited) and Alexander Zuev (Sun Microsystems).

A Special Thank You to Brian Goetz

We were privileged to have Brian Goetz, Oracle’s Java Language Architect and Specification Lead for Java SE 8’s Project Lambda, and co-author of *Java Concurrency in Practice*, do a detailed full-book review. He thoroughly scrutinized every chapter, providing extremely helpful insights and constructive comments. Any remaining faults in the book are our own.
Preface

Well, there you have it! As you read the book, we’d appreciate your comments, criticisms, corrections and suggestions for improvement. Please address all correspondence to:

deitel@deitel.com

We’ll respond promptly. We hope you enjoy working with *Java SE 8 for Programmers, 3/e*, as much as we enjoyed writing it!

*Paul and Harvey Deitel*

About the Authors

*Paul Deitel*, CEO and Chief Technical Officer of Deitel & Associates, Inc., is a graduate of MIT, where he studied Information Technology. He holds the Java Certified Programmer and Java Certified Developer designations, and is an Oracle Java Champion. Through Deitel & Associates, Inc., he has delivered hundreds of programming courses worldwide to clients, including Cisco, IBM, Siemens, Sun Microsystems, Dell, Fidelity, NASA at the Kennedy Space Center, the National Severe Storm Laboratory, White Sands Missile Range, Rogue Wave Software, Boeing, SunGard Higher Education, Nortel Networks, Puma, iRobot, Invensys and many more. He and his co-author, Dr. Harvey M. Deitel, are the world’s best-selling programming-language textbook/professional book/video authors.

*Dr. Harvey Deitel*, Chairman and Chief Strategy Officer of Deitel & Associates, Inc., has over 50 years of experience in the computer field. Dr. Deitel earned B.S. and M.S. degrees in Electrical Engineering from MIT and a Ph.D. in Mathematics from Boston University. He has extensive college teaching experience, including earning tenure and serving as the Chairman of the Computer Science Department at Boston College before founding Deitel & Associates, Inc., in 1991 with his son, Paul. The Deitels’ publications have earned international recognition, with translations published in Japanese, German, Russian, Spanish, French, Polish, Italian, Simplified Chinese, Traditional Chinese, Korean, Portuguese, Greek, Urdu and Turkish. Dr. Deitel has delivered hundreds of programming courses to corporate, academic, government and military clients.

About Deitel® & Associates, Inc.

Deitel & Associates, Inc., founded by Paul Deitel and Harvey Deitel, is an internationally recognized authoring and corporate training organization, specializing in computer programming languages, object technology, mobile app development and Internet and web software technology. The company’s training clients include many of the world’s largest companies, government agencies, branches of the military, and academic institutions. The company offers instructor-led training courses delivered at client sites worldwide on major programming languages and platforms, including *Java™*, Android app development, Objective-C and iOS app development, C++, C, Visual C#®, Visual Basic®, Visual C++®, Python®, object technology, Internet and web programming and a growing list of additional programming and software development courses.

About Deitel® & Associates, Inc.

Through its 39-year publishing partnership with Pearson/Prentice Hall, Deitel & Associates, Inc., publishes leading-edge programming textbooks and professional books in print and a wide range of e-book formats, and LiveLessons video courses. Deitel & Associates, Inc. and the authors can be reached at:

deitel@deitel.com

To learn more about Deitel’s Dive-Into® Series Corporate Training curriculum, visit:

http://www.deitel.com/training

To request a proposal for worldwide on-site, instructor-led training at your organization, e-mail deitel@deitel.com.

Individuals wishing to purchase Deitel books and LiveLessons video training can do so through www.deitel.com. Bulk orders by corporations, the government, the military and academic institutions should be placed directly with Pearson. For more information, visit
