

GLOBAL  
EDITION



# Java™ How to Program

## *Early Objects*

TENTH EDITION

Paul Deitel • Harvey Deitel

ALWAYS LEARNING

PEARSON

# ONLINE ACCESS

Thank you for purchasing a new copy of *Java™ How to Program, Tenth Edition, Early Objects*. Your textbook includes 12 months of prepaid access to the book's Companion Website. This prepaid subscription provides you with full access to the following student support areas:

- VideoNotes (step-by-step video tutorials specifically designed to enhance the programming concepts presented in this textbook)
- Source code
- Premium web chapters and appendices

**Use a coin to scratch off the coating and reveal your student access code.  
Do not use a knife or other sharp object as it may damage the code.**

To access the *Java How to Program, Tenth Edition, Early Objects* Companion Website for the first time, you will need to register online using a computer with an Internet connection and a web browser. The process takes just a couple of minutes and only needs to be completed once.

- 1.** Go to <http://www.pearsonglobaleditions.com/deitel>
- 2.** Click on **Companion Website**.
- 3.** Click on the **Register** button.
- 4.** On the registration page, enter your student access code\* found beneath the scratch-off panel. Do not type the dashes. You can use lower- or uppercase.
- 5.** Follow the on-screen instructions. If you need help at any time during the online registration process, simply click the **Need Help?** icon.
- 6.** Once your personal Login Name and Password are confirmed, you can begin using the *Java How to Program, Tenth Edition, Early Objects* Companion Website!

## **To log in after you have registered:**

You only need to register for this Companion Website once. After that, you can log in any time at <http://www.pearsonglobaleditions.com/deitel> by providing your Login Name and Password when prompted.

\*Important: The access code can only be used once. This subscription is valid for 12 months upon activation and is not transferable. If this access code has already been revealed, it may no longer be valid. If this is the case, you can purchase a subscription at <http://www.pearsonglobaleditions.com/deitel>, by going to the *Java How to Program, Tenth Edition, Early Objects* book and following the on-screen instructions.

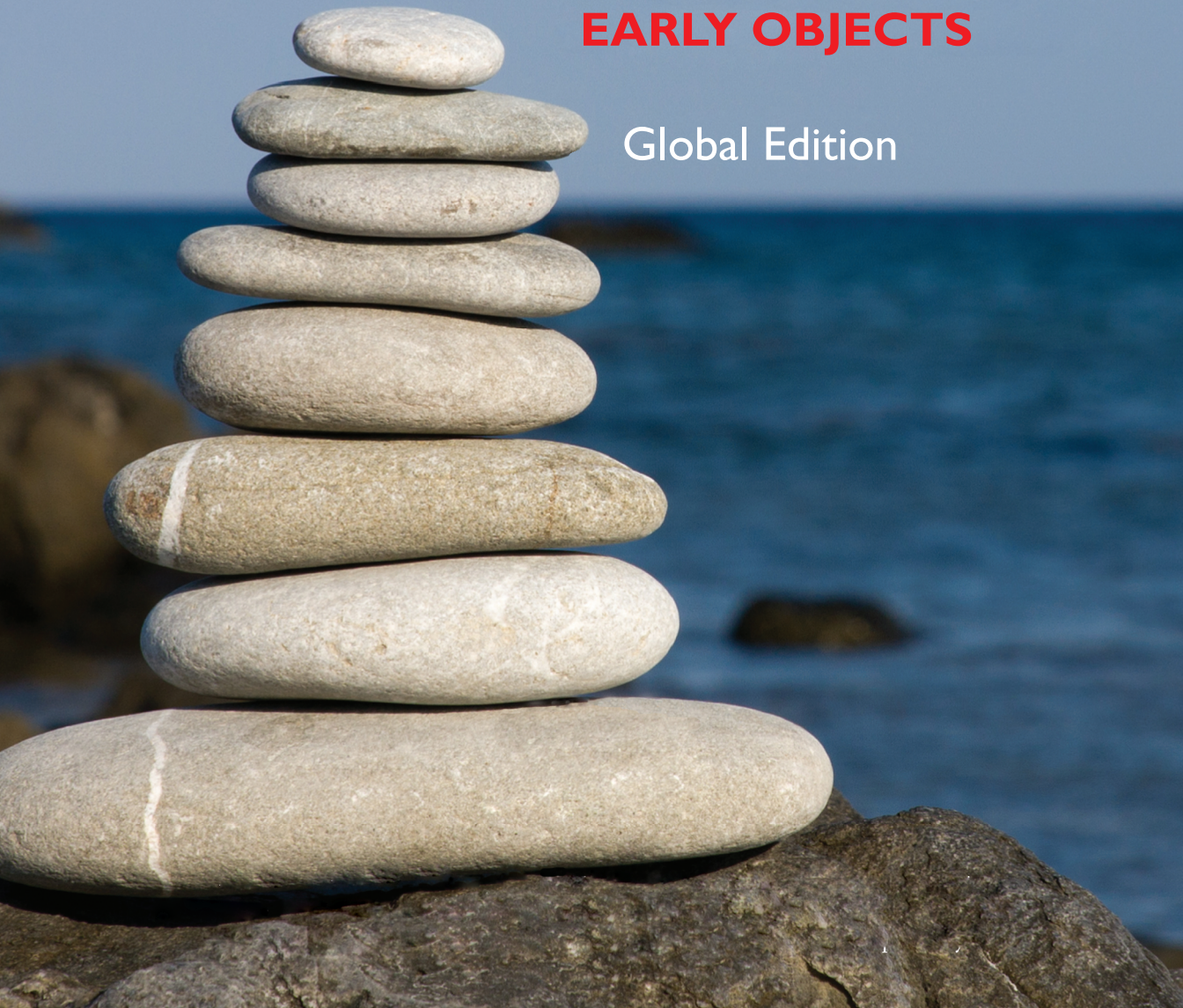
# Java™

## HOW TO PROGRAM

TENTH EDITION

**EARLY OBJECTS**

Global Edition



# Deitel® Series Page

---

## How To Program Series

Android How to Program, 2/E  
C++ How to Program, 9/E  
C How to Program, 7/E  
Java™ How to Program, 10/E  
Java™ How to Program, Late Objects Version, 10/E  
Internet & World Wide Web How to Program, 5/E  
Visual C++® 2008 How to Program, 2/E  
Visual Basic® 2012 How to Program, 6/E  
Visual C#® 2012 How to Program, 5/E

---

## Simply Series

Simply C++: An App-Driven Tutorial Approach  
Simply Java™ Programming: An App-Driven Tutorial Approach  
Simply C#: An App-Driven Tutorial Approach  
Simply Visual Basic® 2010: An App-Driven Approach, 4/E

---

## CourseSmart Web Books\*

[www.deitel.com/books/CourseSmart/](http://www.deitel.com/books/CourseSmart/)  
C++ How to Program, 8/E and 9/E  
Simply C++: An App-Driven Tutorial Approach  
Java™ How to Program, 9/E and 10/E  
Simply Visual Basic® 2010: An App-Driven Approach, 4/E

---

(continued from previous column)

Visual Basic® 2012 How to Program, 6/E  
Visual Basic® 2010 How to Program, 5/E  
Visual C#® 2012 How to Program, 5/E  
Visual C#® 2010 How to Program, 4/E

---

## Deitel® Developer Series

Android for Programmers: An App-Driven Approach, 2/E, Volume 1  
C for Programmers with an Introduction to C11  
C++11 for Programmers  
C# 2012 for Programmers  
Dive Into® iOS 6 for Programmers: An App-Driven Approach  
Java™ for Programmers, 3/E  
JavaScript for Programmers

---

## LiveLessons Video Learning Products

[www.deitel.com/books/LiveLessons/](http://www.deitel.com/books/LiveLessons/)  
Android App Development Fundamentals  
C++ Fundamentals  
Java™ Fundamentals  
C# 2012 Fundamentals  
C# 2010 Fundamentals  
iOS® 6 App Development Fundamentals  
JavaScript Fundamentals  
Visual Basic® Fundamentals

---

To receive updates on Deitel publications, Resource Centers, training courses, partner offers and more, please join the Deitel communities on

- Facebook®—[facebook.com/DeitelFan](https://facebook.com/DeitelFan)
- Twitter®—[@deitel](https://twitter.com/deitel)
- Google+™—[google.com/+DeitelFan](https://google.com/+DeitelFan)
- YouTube™—[youtube.com/DeitelTV](https://youtube.com/DeitelTV)
- LinkedIn®—[linkedin.com/company/deitel-&-associates](https://linkedin.com/company/deitel-&-associates)

and register for the free *Deitel® Buzz Online* e-mail newsletter at:  
[www.deitel.com/newsletter/subscribe.html](http://www.deitel.com/newsletter/subscribe.html)

To communicate with the authors, send e-mail to:  
[deitel@deitel.com](mailto:deitel@deitel.com)

For information on *Dive-Into® Series* on-site seminars offered by Deitel & Associates, Inc. worldwide, write to us at [deitel@deitel.com](mailto:deitel@deitel.com) or visit:  
[www.deitel.com/training/](http://www.deitel.com/training/)

For continuing updates on Pearson/Deitel publications visit:  
[www.deitel.com](http://www.deitel.com)  
[www.pearsonglobaleditions.com/deitel](http://www.pearsonglobaleditions.com/deitel)

Visit the Deitel Resource Centers that will help you master programming languages, software development, Android and iOS app development, and Internet- and web-related topics:  
[www.deitel.com/ResourceCenters.html](http://www.deitel.com/ResourceCenters.html)

\*This product may not be available in all markets. For more details, please visit [www.coursesmart.co.uk](http://www.coursesmart.co.uk) or contact your local Pearson representative.

# Java™

## HOW TO PROGRAM

TENTH EDITION

**EARLY OBJECTS**

Global Edition

**Paul Deitel**

*Deitel & Associates, Inc.*

**Harvey Deitel**

*Deitel & Associates, Inc.*

Global Edition contributions by

**Sherif G. Aly**

**Saleh Al-Hazbi**



**DEITEL®**

Prentice Hall

Boston Columbus Indianapolis New York San Francisco Upper Saddle River  
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto  
Delhi Mexico City Sao Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Editorial Director, ECS: *Marcia Horton*  
Executive Editor: *Tracy Johnson (Dunkelberger)*  
Director of Marketing: *Christy Lesko*  
Marketing Manager: *Yez Alayan*  
Marketing Assistant: *Jon Bryant*  
Director of Program Management: *Erin Gregg*  
Program Management—Team Lead: *Scott Disanno*  
Program Manager: *Carole Snyder*  
Project Management-Team Lead: *Laura Burgess*  
Project Manager: *Robert Engelhardt*  
Procurement Specialist: *Linda Sager*

Cover Design: *Lumina Datamatics, Inc.*  
Permissions Supervisor: *Michael Joyce, Brooks Hill-Wilton*  
Permissions Administrator: *Jenell Forschler*  
Director, Image Asset Services: *Annie Atherton*  
Manager, Visual Research: *Karen Sanatar*  
Cover Art: © *Rafal Olechowski/Shutterstock*  
Media Project Manager: *Renata Butera*  
Head of Learning Asset Acquisition, Global Edition: *Laura Dent*  
Acquisitions Editor, Global Edition: *Subhasree Patra*  
Senior Manufacturing Controller, Global Edition: *Trudy Kimber*  
Project Editor, Global Edition: *Aaditya Bugga*

Credits and acknowledgments borrowed from other sources and reproduced, with permission, in this textbook appear on page 6.

---

**Pearson Education Limited**

Edinburgh Gate  
Harlow  
Essex CM20 2JE  
England

and Associated Companies throughout the world

Visit us on the World Wide Web at:  
[www.pearsonglobaleditions.com](http://www.pearsonglobaleditions.com)

© Pearson Education Limited, 2015

The rights of Paul Deitel and Harvey Deitel to be identified as the authors of this work have been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

*Authorized adaptation from the United States edition, entitled Java How to Program, Early Objects, 10th edition, ISBN 978-0-13-380780-6, by Paul Deitel and Harvey Deitel, published by Pearson Education © 2015.*

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without either the prior written permission of the publisher or a license permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6-10 Kirby Street, London EC1N 8TS.

All trademarks used herein are the property of their respective owners. The use of any trademark in this text does not vest in the author or publisher any trademark ownership rights in such trademarks, nor does the use of such trademarks imply any affiliation with or endorsement of this book by such owners.

ISBN 10: 1-292-01819-4

ISBN 13: 978-1-292-01819-5

**British Library Cataloguing-in-Publication Data**

A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1

Typeset in 10.5 AGaramond-Regular by GEX Publishing Services.

Printed and bound by Courier Kendallville in the United States of America.

---

*To Brian Goetz,  
Oracle's Java Language Architect and  
Specification Lead for Java SE 8's Project Lambda:*

*Your mentorship helped us make a better book,  
Thank you for insisting that we get it right.*

*Paul and Harvey Deitel*

## Trademarks

DEITEL, the double-thumbs-up bug and DIVE INTO are registered trademarks of Deitel and Associates, Inc.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Microsoft and/or its respective suppliers make no representations about the suitability of the information contained in the documents and related graphics published as part of the services for any purpose. All such documents and related graphics are provided “as is” without warranty of any kind. Microsoft and/or its respective suppliers hereby disclaim all warranties and conditions with regard to this information, including all warranties and conditions of merchantability, whether express, implied or statutory, fitness for a particular purpose, title and non-infringement. In no event shall Microsoft and/or its respective suppliers be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of information available from the services.

The documents and related graphics contained herein could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Microsoft and/or its respective suppliers may make improvements and/or changes in the product(s) and/or the program(s) described herein at any time. Partial screen shots may be viewed in full within the software version specified.

Microsoft<sup>®</sup> and Windows<sup>®</sup> are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. Screen shots and icons reprinted with permission from the Microsoft Corporation. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation. UNIX is a registered trademark of The Open Group.

Apache is a trademark of The Apache Software Foundation.

CSS and XML are registered trademarks of the World Wide Web Consortium.

Firefox is a registered trademark of the Mozilla Foundation.

Google is a trademark of Google, Inc.

Mac and OS X are trademarks of Apple Inc., registered in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds. All trademarks are property of their respective owners.

Throughout this book, trademarks are used. Rather than put a trademark symbol in every occurrence of a trademarked name, we state that we are using the names in an editorial fashion only and to the benefit of the trademark owner, with no intention of infringement of the trademark.





# Contents

Chapters 26–34 and Appendices F–N are PDF documents posted online at the book’s Companion Website (located at [www.pearsonglobaleditions.com/deitel](http://www.pearsonglobaleditions.com/deitel)). See the inside front cover for information on accessing the Companion Website.

**Foreword** 23

**Preface** 25

**Before You Begin** 39

**I Introduction to Computers, the Internet and Java** 43

1.1	Introduction	44
1.2	Hardware and Software	46
1.2.1	Moore’s Law	46
1.2.2	Computer Organization	47
1.3	Data Hierarchy	48
1.4	Machine Languages, Assembly Languages and High-Level Languages	51
1.5	Introduction to Object Technology	52
1.5.1	The Automobile as an Object	52
1.5.2	Methods and Classes	53
1.5.3	Instantiation	53
1.5.4	Reuse	53
1.5.5	Messages and Method Calls	53
1.5.6	Attributes and Instance Variables	53
1.5.7	Encapsulation and Information Hiding	54
1.5.8	Inheritance	54
1.5.9	Interfaces	54
1.5.10	Object-Oriented Analysis and Design (OOAD)	54
1.5.11	The UML (Unified Modeling Language)	55
1.6	Operating Systems	55
1.6.1	Windows—A Proprietary Operating System	55
1.6.2	Linux—An Open-Source Operating System	56
1.6.3	Android	56
1.7	Programming Languages	57
1.8	Java	59
1.9	A Typical Java Development Environment	59
1.10	Test-Driving a Java Application	63

1.11	Internet and World Wide Web	67
1.11.1	The Internet: A Network of Networks	68
1.11.2	The World Wide Web: Making the Internet User-Friendly	68
1.11.3	Web Services and Mashups	68
1.11.4	Ajax	69
1.11.5	The Internet of Things	69
1.12	Software Technologies	70
1.13	Keeping Up-to-Date with Information Technologies	72

## 2 Introduction to Java Applications; Input/Output and Operators 76

2.1	Introduction	77
2.2	Your First Program in Java: Printing a Line of Text	77
2.3	Modifying Your First Java Program	83
2.4	Displaying Text with <code>printf</code>	85
2.5	Another Application: Adding Integers	87
2.5.1	<code>import</code> Declarations	87
2.5.2	Declaring Class Addition	88
2.5.3	Declaring and Creating a Scanner to Obtain User Input from the Keyboard	88
2.5.4	Declaring Variables to Store Integers	89
2.5.5	Prompting the User for Input	90
2.5.6	Obtaining an <code>int</code> as Input from the User	90
2.5.7	Prompting for and Inputting a Second <code>int</code>	91
2.5.8	Using Variables in a Calculation	91
2.5.9	Displaying the Result of the Calculation	91
2.5.10	Java API Documentation	91
2.6	Memory Concepts	92
2.7	Arithmetic	93
2.8	Decision Making: Equality and Relational Operators	96
2.9	Wrap-Up	100

## 3 Introduction to Classes, Objects, Methods and Strings III

3.1	Introduction	112
3.2	Instance Variables, <i>set</i> Methods and <i>get</i> Methods	113
3.2.1	Account Class with an Instance Variable, a <i>set</i> Method and a <i>get</i> Method	113
3.2.2	AccountTest Class That Creates and Uses an Object of Class Account	116
3.2.3	Compiling and Executing an App with Multiple Classes	119
3.2.4	Account UML Class Diagram with an Instance Variable and <i>set</i> and <i>get</i> Methods	119
3.2.5	Additional Notes on Class AccountTest	120

3.2.6	Software Engineering with <code>private</code> Instance Variables and <code>public</code> <code>set</code> and <code>get</code> Methods	121
3.3	Primitive Types vs. Reference Types	122
3.4	Account Class: Initializing Objects with Constructors	123
3.4.1	Declaring an Account Constructor for Custom Object Initialization	123
3.4.2	Class <code>AccountTest</code> : Initializing Account Objects When They're Created	124
3.5	Account Class with a Balance; Floating-Point Numbers	126
3.5.1	Account Class with a <code>balance</code> Instance Variable of Type <code>double</code>	127
3.5.2	<code>AccountTest</code> Class to Use Class <code>Account</code>	128
3.6	(Optional) GUI and Graphics Case Study: Using Dialog Boxes	132
3.7	Wrap-Up	135

## 4 Control Statements: Part 1; Assignment, ++ and -- Operators 143

4.1	Introduction	144
4.2	Algorithms	144
4.3	Pseudocode	145
4.4	Control Structures	145
4.5	<code>if</code> Single-Selection Statement	147
4.6	<code>if...else</code> Double-Selection Statement	148
4.7	Student Class: Nested <code>if...else</code> Statements	153
4.8	<code>while</code> Repetition Statement	155
4.9	Formulating Algorithms: Counter-Controlled Repetition	157
4.10	Formulating Algorithms: Sentinel-Controlled Repetition	161
4.11	Formulating Algorithms: Nested Control Statements	168
4.12	Compound Assignment Operators	173
4.13	Increment and Decrement Operators	173
4.14	Primitive Types	176
4.15	(Optional) GUI and Graphics Case Study: Creating Simple Drawings	177
4.16	Wrap-Up	181

## 5 Control Statements: Part 2; Logical Operators 194

5.1	Introduction	195
5.2	Essentials of Counter-Controlled Repetition	195
5.3	<code>for</code> Repetition Statement	197
5.4	Examples Using the <code>for</code> Statement	201
5.5	<code>do...while</code> Repetition Statement	205
5.6	<code>switch</code> Multiple-Selection Statement	207
5.7	Class <code>AutoPolicy</code> Case Study: Strings in <code>switch</code> Statements	213
5.8	<code>break</code> and <code>continue</code> Statements	216
5.9	Logical Operators	218
5.10	Structured Programming Summary	224
5.11	(Optional) GUI and Graphics Case Study: Drawing Rectangles and Ovals	229
5.12	Wrap-Up	232

<b>6</b>	<b>Methods: A Deeper Look</b>	<b>242</b>
6.1	Introduction	243
6.2	Program Modules in Java	243
6.3	static Methods, static Fields and Class Math	245
6.4	Declaring Methods with Multiple Parameters	247
6.5	Notes on Declaring and Using Methods	250
6.6	Method-Call Stack and Stack Frames	251
6.7	Argument Promotion and Casting	252
6.8	Java API Packages	253
6.9	Case Study: Secure Random-Number Generation	255
6.10	Case Study: A Game of Chance; Introducing enum Types	260
6.11	Scope of Declarations	264
6.12	Method Overloading	267
6.13	(Optional) GUI and Graphics Case Study: Colors and Filled Shapes	269
6.14	Wrap-Up	272
<b>7</b>	<b>Arrays and ArrayLists</b>	<b>285</b>
7.1	Introduction	286
7.2	Arrays	287
7.3	Declaring and Creating Arrays	288
7.4	Examples Using Arrays	289
7.4.1	Creating and Initializing an Array	289
7.4.2	Using an Array Initializer	290
7.4.3	Calculating the Values to Store in an Array	291
7.4.4	Summing the Elements of an Array	293
7.4.5	Using Bar Charts to Display Array Data Graphically	293
7.4.6	Using the Elements of an Array as Counters	295
7.4.7	Using Arrays to Analyze Survey Results	296
7.5	Exception Handling: Processing the Incorrect Response	298
7.5.1	The try Statement	298
7.5.2	Executing the catch Block	298
7.5.3	toString Method of the Exception Parameter	299
7.6	Case Study: Card Shuffling and Dealing Simulation	299
7.7	Enhanced for Statement	304
7.8	Passing Arrays to Methods	305
7.9	Pass-By-Value vs. Pass-By-Reference	307
7.10	Case Study: Class GradeBook Using an Array to Store Grades	308
7.11	Multidimensional Arrays	314
7.12	Case Study: Class GradeBook Using a Two-Dimensional Array	317
7.13	Variable-Length Argument Lists	323
7.14	Using Command-Line Arguments	325
7.15	Class Arrays	327
7.16	Introduction to Collections and Class ArrayList	329
7.17	(Optional) GUI and Graphics Case Study: Drawing Arcs	333
7.18	Wrap-Up	336

<b>8</b>	<b>Classes and Objects: A Deeper Look</b>	<b>357</b>
8.1	Introduction	358
8.2	Time Class Case Study	358
8.3	Controlling Access to Members	363
8.4	Referring to the Current Object's Members with the <code>this</code> Reference	364
8.5	Time Class Case Study: Overloaded Constructors	366
8.6	Default and No-Argument Constructors	372
8.7	Notes on <i>Set</i> and <i>Get</i> Methods	372
8.8	Composition	374
8.9	<code>enum</code> Types	377
8.10	Garbage Collection	379
8.11	<code>static</code> Class Members	380
8.12	<code>static</code> Import	384
8.13	<code>final</code> Instance Variables	385
8.14	Package Access	386
8.15	Using <code>BigDecimal</code> for Precise Monetary Calculations	387
8.16	(Optional) GUI and Graphics Case Study: Using Objects with Graphics	390
8.17	Wrap-Up	394
<b>9</b>	<b>Object-Oriented Programming: Inheritance</b>	<b>402</b>
9.1	Introduction	403
9.2	Superclasses and Subclasses	404
9.3	<code>protected</code> Members	406
9.4	Relationship Between Superclasses and Subclasses	407
9.4.1	Creating and Using a <code>CommissionEmployee</code> Class	407
9.4.2	Creating and Using a <code>BasePlusCommissionEmployee</code> Class	413
9.4.3	Creating a <code>CommissionEmployee–BasePlusCommissionEmployee</code> Inheritance Hierarchy	418
9.4.4	<code>CommissionEmployee–BasePlusCommissionEmployee</code> Inheritance Hierarchy Using <code>protected</code> Instance Variables	421
9.4.5	<code>CommissionEmployee–BasePlusCommissionEmployee</code> Inheritance Hierarchy Using <code>private</code> Instance Variables	424
9.5	Constructors in Subclasses	429
9.6	Class Object	429
9.7	(Optional) GUI and Graphics Case Study: Displaying Text and Images Using Labels	430
9.8	Wrap-Up	433
<b>10</b>	<b>Object-Oriented Programming: Polymorphism and Interfaces</b>	<b>437</b>
10.1	Introduction	438
10.2	Polymorphism Examples	440
10.3	Demonstrating Polymorphic Behavior	441
10.4	Abstract Classes and Methods	443

## 12 Contents

10.5	Case Study: Payroll System Using Polymorphism	446
10.5.1	Abstract Superclass Employee	447
10.5.2	Concrete Subclass SalariedEmployee	449
10.5.3	Concrete Subclass HourlyEmployee	451
10.5.4	Concrete Subclass CommissionEmployee	453
10.5.5	Indirect Concrete Subclass BasePlusCommissionEmployee	455
10.5.6	Polymorphic Processing, Operator instanceof and Downcasting	456
10.6	Allowed Assignments Between Superclass and Subclass Variables	461
10.7	final Methods and Classes	461
10.8	A Deeper Explanation of Issues with Calling Methods from Constructors	462
10.9	Creating and Using Interfaces	463
10.9.1	Developing a Payable Hierarchy	464
10.9.2	Interface Payable	465
10.9.3	Class Invoice	466
10.9.4	Modifying Class Employee to Implement Interface Payable	468
10.9.5	Modifying Class SalariedEmployee for Use in the Payable Hierarchy	470
10.9.6	Using Interface Payable to Process Invoices and Employees Polymorphically	472
10.9.7	Some Common Interfaces of the Java API	473
10.10	Java SE 8 Interface Enhancements	474
10.10.1	default Interface Methods	474
10.10.2	static Interface Methods	475
10.10.3	Functional Interfaces	475
10.11	(Optional) GUI and Graphics Case Study: Drawing with Polymorphism	475
10.12	Wrap-Up	478

## 11 Exception Handling: A Deeper Look 483

11.1	Introduction	484
11.2	Example: Divide by Zero without Exception Handling	485
11.3	Example: Handling ArithmeticExceptions and InputMismatchExceptions	487
11.4	When to Use Exception Handling	493
11.5	Java Exception Hierarchy	493
11.6	finally Block	496
11.7	Stack Unwinding and Obtaining Information from an Exception Object	501
11.8	Chained Exceptions	503
11.9	Declaring New Exception Types	506
11.10	Preconditions and Postconditions	507
11.11	Assertions	507
11.12	try-with-Resources: Automatic Resource Deallocation	509
11.13	Wrap-Up	509

## 12 GUI Components: Part I 515

12.1	Introduction	516
------	--------------	-----

12.2	Java's Nimbus Look-and-Feel	517
12.3	Simple GUI-Based Input/Output with JOptionPane	518
12.4	Overview of Swing Components	521
12.5	Displaying Text and Images in a Window	523
12.6	Text Fields and an Introduction to Event Handling with Nested Classes	527
12.7	Common GUI Event Types and Listener Interfaces	533
12.8	How Event Handling Works	535
12.9	JButton	537
12.10	Buttons That Maintain State	540
	12.10.1 JCheckBox	541
	12.10.2 JRadioButton	543
12.11	JComboBox; Using an Anonymous Inner Class for Event Handling	546
12.12	JList	550
12.13	Multiple-Selection Lists	553
12.14	Mouse Event Handling	555
12.15	Adapter Classes	560
12.16	JPanel Subclass for Drawing with the Mouse	564
12.17	Key Event Handling	567
12.18	Introduction to Layout Managers	570
	12.18.1 FlowLayout	572
	12.18.2 BorderLayout	574
	12.18.3 GridLayout	578
12.19	Using Panels to Manage More Complex Layouts	580
12.20	JTextArea	581
12.21	Wrap-Up	584

## **13 Graphics and Java 2D** **597**

13.1	Introduction	598
13.2	Graphics Contexts and Graphics Objects	600
13.3	Color Control	601
13.4	Manipulating Fonts	608
13.5	Drawing Lines, Rectangles and Ovals	613
13.6	Drawing Arcs	617
13.7	Drawing Polygons and Polylines	620
13.8	Java 2D API	623
13.9	Wrap-Up	630

## **14 Strings, Characters and Regular Expressions** **638**

14.1	Introduction	639
14.2	Fundamentals of Characters and Strings	639
14.3	Class String	640
	14.3.1 String Constructors	640
	14.3.2 String Methods length, charAt and getChars	641
	14.3.3 Comparing Strings	642

## 14 Contents

14.3.4	Locating Characters and Substrings in Strings	647
14.3.5	Extracting Substrings from Strings	649
14.3.6	Concatenating Strings	650
14.3.7	Miscellaneous String Methods	650
14.3.8	String Method <code>valueOf</code>	652
14.4	Class <code>StringBuilder</code>	653
14.4.1	<code>StringBuilder</code> Constructors	654
14.4.2	<code>StringBuilder</code> Methods <code>length</code> , <code>capacity</code> , <code>setLength</code> and <code>ensureCapacity</code>	654
14.4.3	<code>StringBuilder</code> Methods <code>charAt</code> , <code>setCharAt</code> , <code>getChars</code> and <code>reverse</code>	656
14.4.4	<code>StringBuilder</code> <code>append</code> Methods	657
14.4.5	<code>StringBuilder</code> Insertion and Deletion Methods	659
14.5	Class <code>Character</code>	660
14.6	Tokenizing Strings	665
14.7	Regular Expressions, Class <code>Pattern</code> and Class <code>Matcher</code>	666
14.8	Wrap-Up	675

## 15 Files, Streams and Object Serialization 686

15.1	Introduction	687
15.2	Files and Streams	687
15.3	Using NIO Classes and Interfaces to Get File and Directory Information	689
15.4	Sequential-Access Text Files	693
15.4.1	Creating a Sequential-Access Text File	693
15.4.2	Reading Data from a Sequential-Access Text File	697
15.4.3	Case Study: A Credit-Inquiry Program	699
15.4.4	Updating Sequential-Access Files	703
15.5	Object Serialization	704
15.5.1	Creating a Sequential-Access File Using Object Serialization	705
15.5.2	Reading and Deserializing Data from a Sequential-Access File	710
15.6	Opening Files with <code>JFileChooser</code>	712
15.7	(Optional) Additional <code>java.io</code> Classes	715
15.7.1	Interfaces and Classes for Byte-Based Input and Output	715
15.7.2	Interfaces and Classes for Character-Based Input and Output	717
15.8	Wrap-Up	718

## 16 Generic Collections 726

16.1	Introduction	727
16.2	Collections Overview	727
16.3	Type-Wrapper Classes	729
16.4	Autoboxing and Auto-Unboxing	729
16.5	Interface <code>Collection</code> and Class <code>Collections</code>	729
16.6	Lists	730
16.6.1	<code>ArrayList</code> and <code>Iterator</code>	731
16.6.2	<code>LinkedList</code>	733



16.7	Collections Methods	738
16.7.1	Method <code>sort</code>	739
16.7.2	Method <code>shuffle</code>	742
16.7.3	Methods <code>reverse</code> , <code>fill</code> , <code>copy</code> , <code>max</code> and <code>min</code>	744
16.7.4	Method <code>binarySearch</code>	746
16.7.5	Methods <code>addAll</code> , <code>frequency</code> and <code>disjoint</code>	748
16.8	Stack Class of Package <code>java.util</code>	750
16.9	Class <code>PriorityQueue</code> and Interface <code>Queue</code>	752
16.10	Sets	753
16.11	Maps	756
16.12	<code>Properties</code> Class	760
16.13	Synchronized Collections	763
16.14	Unmodifiable Collections	763
16.15	Abstract Implementations	764
16.16	Wrap-Up	764

## **17** Java SE 8 Lambdas and Streams **771**

17.1	Introduction	772
17.2	Functional Programming Technologies Overview	773
17.2.1	Functional Interfaces	774
17.2.2	Lambda Expressions	775
17.2.3	Streams	776
17.3	<code>IntStream</code> Operations	778
17.3.1	Creating an <code>IntStream</code> and Displaying Its Values with the <code>forEach</code> Terminal Operation	780
17.3.2	Terminal Operations <code>count</code> , <code>min</code> , <code>max</code> , <code>sum</code> and <code>average</code>	781
17.3.3	Terminal Operation <code>reduce</code>	781
17.3.4	Intermediate Operations: Filtering and Sorting <code>IntStream</code> Values	783
17.3.5	Intermediate Operation: Mapping	784
17.3.6	Creating Streams of ints with <code>IntStream</code> Methods <code>range</code> and <code>rangeClosed</code>	785
17.4	<code>Stream&lt;Integer&gt;</code> Manipulations	785
17.4.1	Creating a <code>Stream&lt;Integer&gt;</code>	786
17.4.2	Sorting a <code>Stream</code> and Collecting the Results	787
17.4.3	Filtering a <code>Stream</code> and Storing the Results for Later Use	787
17.4.4	Filtering and Sorting a <code>Stream</code> and Collecting the Results	787
17.4.5	Sorting Previously Collected Results	787
17.5	<code>Stream&lt;String&gt;</code> Manipulations	788
17.5.1	Mapping Strings to Uppercase Using a Method Reference	789
17.5.2	Filtering Strings Then Sorting Them in Case-Insensitive Ascending Order	790
17.5.3	Filtering Strings Then Sorting Them in Case-Insensitive Descending Order	790
17.6	<code>Stream&lt;Employee&gt;</code> Manipulations	790
17.6.1	Creating and Displaying a <code>List&lt;Employee&gt;</code>	792

## 16 Contents

17.6.2	Filtering Employees with Salaries in a Specified Range	793
17.6.3	Sorting Employees By Multiple Fields	794
17.6.4	Mapping Employees to Unique Last Name Strings	796
17.6.5	Grouping Employees By Department	797
17.6.6	Counting the Number of Employees in Each Department	798
17.6.7	Summing and Averaging Employee Salaries	798
17.7	Creating a Stream<String> from a File	800
17.8	Generating Streams of Random Values	803
17.9	Lambda Event Handlers	805
17.10	Additional Notes on Java SE 8 Interfaces	805
17.11	Java SE 8 and Functional Programming Resources	806
17.12	Wrap-Up	806

## 18 Recursion 818

18.1	Introduction	819
18.2	Recursion Concepts	820
18.3	Example Using Recursion: Factorials	821
18.4	Reimplementing Class <code>FactorialCalculator</code> Using Class <code>BigInteger</code>	823
18.5	Example Using Recursion: Fibonacci Series	825
18.6	Recursion and the Method-Call Stack	828
18.7	Recursion vs. Iteration	829
18.8	Towers of Hanoi	831
18.9	Fractals	833
18.9.1	Koch Curve Fractal	833
18.9.2	(Optional) Case Study: Lo Feather Fractal	834
18.10	Recursive Backtracking	843
18.11	Wrap-Up	844

## 19 Searching, Sorting and Big O 852

19.1	Introduction	853
19.2	Linear Search	854
19.3	Big O Notation	856
19.3.1	$O(1)$ Algorithms	856
19.3.2	$O(n)$ Algorithms	857
19.3.3	$O(n^2)$ Algorithms	857
19.3.4	Big O of the Linear Search	858
19.4	Binary Search	858
19.4.1	Binary Search Implementation	859
19.4.2	Efficiency of the Binary Search	862
19.5	Sorting Algorithms	862
19.6	Selection Sort	863
19.6.1	Selection Sort Implementation	863
19.6.2	Efficiency of the Selection Sort	866
19.7	Insertion Sort	866

19.7.1	Insertion Sort Implementation	867
19.7.2	Efficiency of the Insertion Sort	869
19.8	Merge Sort	869
19.8.1	Merge Sort Implementation	870
19.8.2	Efficiency of the Merge Sort	874
19.9	Big O Summary for This Chapter's Searching and Sorting Algorithms	875
19.10	Wrap-Up	876

## 20 Generic Classes and Methods 881

20.1	Introduction	882
20.2	Motivation for Generic Methods	882
20.3	Generic Methods: Implementation and Compile-Time Translation	884
20.4	Additional Compile-Time Translation Issues: Methods That Use a Type Parameter as the Return Type	887
20.5	Overloading Generic Methods	890
20.6	Generic Classes	891
20.7	Raw Types	898
20.8	Wildcards in Methods That Accept Type Parameters	902
20.9	Wrap-Up	906

## 21 Custom Generic Data Structures 911

21.1	Introduction	912
21.2	Self-Referential Classes	913
21.3	Dynamic Memory Allocation	913
21.4	Linked Lists	914
21.4.1	Singly Linked Lists	914
21.4.2	Implementing a Generic List Class	915
21.4.3	Generic Classes ListNode and List	920
21.4.4	Class ListTest	920
21.4.5	List Method insertAtFront	920
21.4.6	List Method insertAtBack	921
21.4.7	List Method removeFromFront	922
21.4.8	List Method removeFromBack	923
21.4.9	List Method print	924
21.4.10	Creating Your Own Packages	924
21.5	Stacks	928
21.6	Queues	932
21.7	Trees	935
21.8	Wrap-Up	942

## 22 GUI Components: Part 2 953

22.1	Introduction	954
22.2	JSlider	954

22.3	Understanding Windows in Java	958
22.4	Using Menus with Frames	959
22.5	JPopupMenu	967
22.6	Pluggable Look-and-Feel	970
22.7	JDesktopPane and JInternalFrame	975
22.8	JTabbedPane	978
22.9	BoxLayout Layout Manager	980
22.10	GridBagLayout Layout Manager	984
22.11	Wrap-Up	994

## 23 Concurrency 999

23.1	Introduction	1000
23.2	Thread States and Life Cycle	1002
23.2.1	<i>New</i> and <i>Runnable</i> States	1003
23.2.2	<i>Waiting</i> State	1003
23.2.3	<i>Timed Waiting</i> State	1003
23.2.4	<i>Blocked</i> State	1003
23.2.5	<i>Terminated</i> State	1003
23.2.6	Operating-System View of the <i>Runnable</i> State	1004
23.2.7	Thread Priorities and Thread Scheduling	1004
23.2.8	Indefinite Postponement and Deadlock	1005
23.3	Creating and Executing Threads with the Executor Framework	1005
23.4	Thread Synchronization	1009
23.4.1	Immutable Data	1010
23.4.2	Monitors	1010
23.4.3	Unsynchronized Mutable Data Sharing	1011
23.4.4	Synchronized Mutable Data Sharing—Making Operations Atomic	1016
23.5	Producer/Consumer Relationship without Synchronization	1018
23.6	Producer/Consumer Relationship: ArrayBlockingQueue	1026
23.7	(Advanced) Producer/Consumer Relationship with synchronized, wait, notify and notifyAll	1029
23.8	(Advanced) Producer/Consumer Relationship: Bounded Buffers	1036
23.9	(Advanced) Producer/Consumer Relationship: The Lock and Condition Interfaces	1044
23.10	Concurrent Collections	1051
23.11	Multithreading with GUI: SwingWorker	1053
23.11.1	Performing Computations in a Worker Thread: Fibonacci Numbers	1054
23.11.2	Processing Intermediate Results: Sieve of Eratosthenes	1060
23.12	sort and parallelSort Timings with the Java SE 8 Date/Time API	1067
23.13	Java SE 8: Sequential vs. Parallel Streams	1069
23.14	(Advanced) Interfaces Callable and Future	1072
23.15	(Advanced) Fork/Join Framework	1076
23.16	Wrap-Up	1076

<b>24</b>	<b>Accessing Databases with JDBC</b>	<b>1087</b>
24.1	Introduction	1088
24.2	Relational Databases	1089
24.3	A books Database	1090
24.4	SQL	1094
24.4.1	Basic SELECT Query	1094
24.4.2	WHERE Clause	1095
24.4.3	ORDER BY Clause	1097
24.4.4	Merging Data from Multiple Tables: INNER JOIN	1098
24.4.5	INSERT Statement	1100
24.4.6	UPDATE Statement	1101
24.4.7	DELETE Statement	1102
24.5	Setting up a Java DB Database	1102
24.5.1	Creating the Chapter's Databases on Windows	1103
24.5.2	Creating the Chapter's Databases on Mac OS X	1104
24.5.3	Creating the Chapter's Databases on Linux	1105
24.6	Manipulating Databases with JDBC	1105
24.6.1	Connecting to and Querying a Database	1105
24.6.2	Querying the books Database	1109
24.7	RowSet Interface	1122
24.8	PreparedStatement	1124
24.9	Stored Procedures	1140
24.10	Transaction Processing	1140
24.11	Wrap-Up	1141
<b>25</b>	<b>JavaFX GUI: Part I</b>	<b>1149</b>
25.1	Introduction	1150
25.2	JavaFX Scene Builder and the NetBeans IDE	1151
25.3	JavaFX App Window Structure	1152
25.4	<b>Welcome App</b> —Displaying Text and an Image	1153
25.4.1	Creating the App's Project	1153
25.4.2	NetBeans <b>Projects</b> Window—Viewing the Project Contents	1155
25.4.3	Adding an Image to the Project	1156
25.4.4	Opening JavaFX Scene Builder from NetBeans	1156
25.4.5	Changing to a VBox Layout Container	1157
25.4.6	Configuring the VBox Layout Container	1158
25.4.7	Adding and Configuring a Label	1158
25.4.8	Adding and Configuring an ImageView	1158
25.4.9	Running the Welcome App	1159
25.5	<b>Tip Calculator App</b> —Introduction to Event Handling	1160
25.5.1	Test-Driving the <b>Tip Calculator</b> App	1161
25.5.2	Technologies Overview	1161
25.5.3	Building the App's GUI	1164
25.5.4	TipCalculator Class	1168
25.5.5	TipCalculatorController Class	1170

25.6	Features Covered in the Online JavaFX Chapters	1175
25.7	Wrap-Up	1176

## **Chapters on the Web** **I 183**

### **A** Operator Precedence Chart **I 185**

### **B** ASCII Character Set **I 187**

### **C** Keywords and Reserved Words **I 188**

### **D** Primitive Types **I 189**

### **E** Using the Debugger **I 190**

E.1	Introduction	1191
E.2	Breakpoints and the run, stop, cont and print Commands	1191
E.3	The print and set Commands	1195
E.4	Controlling Execution Using the step, step up and next Commands	1197
E.5	The watch Command	1200
E.6	The clear Command	1202
E.7	Wrap-Up	1204

## **Appendices on the Web** **I 207**

## **Index** **I 209**

## **Online Chapters and Appendices**

Chapters 26–34 and Appendices F–N are PDF documents posted online at the book’s Companion Website (located at [www.pearsonglobaleditions.com/deitel](http://www.pearsonglobaleditions.com/deitel)). See the inside front cover for information on accessing the Companion Website.

## **26** JavaFX GUI: Part 2

## **27** JavaFX Graphics and Multimedia

## **28** Networking

## **29** Java Persistence API (JPA)

## **30** JavaServer™ Faces Web Apps: Part I

- 31** JavaServer™ Faces Web Apps: Part 2
- 32** REST-Based Web Services
- 33** (Optional) ATM Case Study, Part 1: Object-Oriented Design with the UML
- 34** (Optional) ATM Case Study, Part 2: Implementing an Object-Oriented Design
- F** Using the Java API Documentation
- G** Creating Documentation with javadoc
- H** Unicode®
- I** Formatted Output
- J** Number Systems
- K** Bit Manipulation
- L** Labeled break and continue Statements
- M** UML 2: Additional Diagram Types
- N** Design Patterns







## Foreword

I've been enamored with Java even prior to its 1.0 release in 1995, and have subsequently been a Java developer, author, speaker, teacher and Oracle Java Technology Ambassador. In this journey, it has been my privilege to call Paul Deitel a colleague, and to often leverage and recommend his *Java How To Program* book. In its many editions, this book has proven to be a great text for college and professional courses that I and others have developed to teach the Java programming language.

One of the qualities that makes this book a great resource is its thorough and insightful coverage of Java concepts, including those introduced recently in Java SE 8. Another useful quality is its treatment of concepts and practices essential to effective software development.

As a long-time fan of this book, I'd like to point out some of the features of this tenth edition about which I'm most excited:

- An ambitious new chapter on Java lambda expressions and streams. This chapter starts out with a primer on functional programming, introducing Java lambda expressions and how to use streams to perform functional programming tasks on collections.
- Although concurrency has been addressed since the first edition of the book, it is increasingly important because of multi-core architectures. There are timing examples—using the new Date/Time API classes introduced in Java SE 8—in the concurrency chapter that show the performance improvements with multi-core over single-core.
- JavaFX is Java's GUI/graphics/multimedia technology moving forward, so it is nice to see a three-chapter treatment of JavaFX in the Deitel live-code pedagogic style. One of these chapters is in the printed book and the other two are online.

Please join me in congratulating Paul and Harvey Deitel on their latest edition of a wonderful resource for computer science students and software developers alike!

James L. Weaver  
Java Technology Ambassador  
Oracle Corporation





# Preface

*“The chief merit of language is clearness...”*

—Galen

Welcome to the Java programming language and *Java How to Program, Tenth Edition*! This book presents leading-edge computing technologies for students, instructors and software developers. It’s appropriate for introductory academic and professional course sequences based on the curriculum recommendations of the ACM and the IEEE, and for AP Computer Science exam preparation. Please read this Preface carefully, we’ve mentioned important details about the book.

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<sup>®</sup>, OS X<sup>®</sup> and Linux<sup>®</sup>. Each complete code example is accompanied by live sample executions.

## *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](mailto:deitel@deitel.com)

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

<http://www.pearsonglobaleditions.com/deitel>

subscribe to the *Deitel<sup>®</sup> Buzz Online* newsletter at

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

and join the Deitel social networking communities on

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

## *Source Code and VideoNotes*

All the source code is available at the book’s Companion Website (which also contains extensive VideoNotes):

<http://www.pearsonglobaleditions.com/deitel>

## Modular Organization<sup>1</sup>

*Java How to Program, 10/e*, is appropriate for programming courses at various levels, most notably CS 1 and CS 2 courses and introductory course sequences in related disciplines. The book's modular organization helps instructors plan their syllabi:

### *Introduction*

- Chapter 1, Introduction to Computers, the Internet and Java
- Chapter 2, Introduction to Java Applications; Input/Output and Operators
- Chapter 3, Introduction to Classes, Objects, Methods and Strings

### *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 and Object-Oriented Design*

- 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
- (Online) Chapter 33, ATM Case Study, Part 1: Object-Oriented Design with the UML
- (Online) Chapter 34, ATM Case Study Part 2: Implementing an Object-Oriented Design

### *Swing Graphical User Interfaces and Java 2D Graphics*

- Chapter 12, GUI Components: Part 1
- Chapter 13, Graphics and Java 2D
- Chapter 22, GUI Components: Part 2

### *Data Structures, Collections, Lambdas and Streams*

- Chapter 16, Generic Collections
- Chapter 17, Java SE 8 Lambdas and Streams
- Chapter 18, Recursion
- Chapter 19, Searching, Sorting and Big O
- Chapter 20, Generic Classes and Methods
- Chapter 21, Custom Generic Data Structures

---

1. The online chapters will be available on the book's Companion Website for Autumn 2014 classes.

*Concurrency; Networking*

- Chapter 23, Concurrency
- (Online) Chapter 28, Networking

*JavaFX Graphical User Interfaces, Graphics and Multimedia*

- Chapter 25, JavaFX GUI: Part 1
- (Online) Chapter 26, JavaFX GUI: Part 2
- (Online) Chapter 27, JavaFX Graphics and Multimedia

*Database-Driven Desktop and Web Development*

- Chapter 24, Accessing Databases with JDBC
- (Online) Chapter 29, Java Persistence API (JPA)
- (Online) Chapter 30, JavaServer™ Faces Web Apps: Part 1
- (Online) Chapter 31, JavaServer™ Faces Web Apps: Part 2
- (Online) Chapter 32, REST-Based Web Services

**New and Updated Features**

Here are the updates we've made for *Java How to Program, 10/e*:

*Java Standard Edition: Java SE 7 and the New Java SE 8*

- *Easy to use with Java SE 7 or Java SE 8.* To meet the needs of our audiences, we designed the book for college and professional courses based on Java SE 7, Java SE 8 or a mixture of both. The Java SE 8 features are covered in optional, easy-to-include-or-omit sections. The new Java SE 8 capabilities can dramatically improve the programming process. Figure 1 lists some new Java SE 8 features that we cover.

## 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. 1** | Some new Java SE 8 features.

- *Java SE 8 lambdas, streams, and interfaces with default and static methods.* The most significant new features in JavaSE 8 are lambdas and complementary technologies, which we cover in detail in the optional Chapter 17 and optional sections marked “Java SE 8” in later chapters. 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. You’ll see that functional programming complements object-oriented programming. After you read Chapter 17, you’ll be able to cleverly reimplement many of the Java SE 7 examples throughout the book (Fig. 2).

Pre-Java-SE-8 topics	Corresponding Java SE 8 discussions and examples
Chapter 7, Arrays and ArrayLists	Sections 17.3–17.4 introduce basic lambda and streams capabilities that process one-dimensional arrays.
Chapter 10, Object-Oriented Programming: Polymorphism and Interfaces	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.
Chapters 12 and 22, GUI Components: Part 1 and 2, respectively	Section 17.9 shows how to use a lambda to implement a Swing event-listener functional interface.
Chapter 14, Strings, Characters and Regular Expressions	Section 17.5 shows how to use lambdas and streams to process collections of String objects.
Chapter 15, Files, Streams and Object Serialization	Section 17.7 shows how to use lambdas and streams to process lines of text from a file.
Chapter 23, Concurrency	Shows that functional programs are easier to parallelize so that they can take advantage of multi-core architectures to enhance performance. Demonstrates parallel stream processing. Shows that Arrays method parallelSort improves performance on multi-core architectures when sorting large arrays.
Chapter 25, JavaFX GUI: Part 1	Section 25.5.5 shows how to use a lambda to implement a JavaFX event-listener functional interface.

**Fig. 2** | Java SE 8 lambdas and streams discussions and examples.

- *Java SE 7’s try-with-resources statement and the AutoCloseable Interface.* AutoCloseable objects reduce the likelihood of resource leaks when you use them with the try-with-resources statement, which automatically closes the AutoCloseable objects. In this edition, we use try-with-resources and AutoCloseable 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 as appropriate for an introductory textbook.

<http://bit.ly/CERTOracleSecureJava>

See the Secure Java Programming section of this Preface for more information 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.pearsonglobaleditions.com/deitel`

### *Swing and JavaFX GUI, Graphics and Multimedia*

- *Swing GUI and Java 2D graphics.* Java's Swing GUI is discussed in the optional GUI and graphics sections in Chapters 3–10 and in Chapters 12 and 22. Swing is now in maintenance mode—Oracle has stopped development and will provide only bug fixes going forward, however it will remain part of Java and is still widely used. Chapter 13 discusses Java 2D graphics.
- *JavaFX GUI, graphics and multimedia.* Java's GUI, graphics and multimedia API going forward is JavaFX. In Chapter 25, we use JavaFX 2.2 (released in 2012) with Java SE 7. Our online Chapters 26 and 27—located on the book's companion website (see the inside front cover of this book)—present additional JavaFX GUI features and introduce JavaFX graphics and multimedia in the context of Java FX 8 and Java SE 8. In Chapters 25–27 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 any of the Java IDEs.
- *Scalable GUI and graphics presentation.* Instructors teaching introductory courses have a broad choice of the amount of GUI, graphics and multimedia to cover—from none at all, to optional introductory sections in the early chapters, to a deep treatment of Swing GUI and Java 2D graphics in Chapters 12, 13 and 22, and a deep treatment of JavaFX GUI, graphics and multimedia in Chapter 25 and online Chapters 26–27.

### *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 23, 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. Fi-

nally, we added a Java SE 8 `CompletableFuture` example that demonstrates sequential and parallel execution of long-running calculations.

- ***SwingWorker class.*** We use class `SwingWorker` to create multithreaded user interfaces. In online Chapter 26, we show how JavaFX handles concurrency.
- ***Concurrency is challenging.*** Programming concurrent applications is difficult and error-prone. There’s a great variety of concurrency features. We point out the ones that most people 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 25.

### *Object Technology*

- ***Object-oriented programming and design.*** We use an *early objects* approach, introducing the basic concepts and terminology of object technology in Chapter 1. Students develop their first customized classes and objects in Chapter 3. Presenting objects and classes early gets students “thinking about objects” immediately and mastering these concepts more thoroughly. [For courses that require a late-objects approach, consider *Java How to Program, 10/e, Late Objects Version.*]
- ***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.
- ***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. The chapter’s rich selection of exercises includes a substantial project on building your own computer through the technique of software simulation. Chapter 21 includes a follow-on project on building your own compiler that can compile high-level language programs into machine language code that will execute on your computer simulator.