

GLOBAL
EDITION

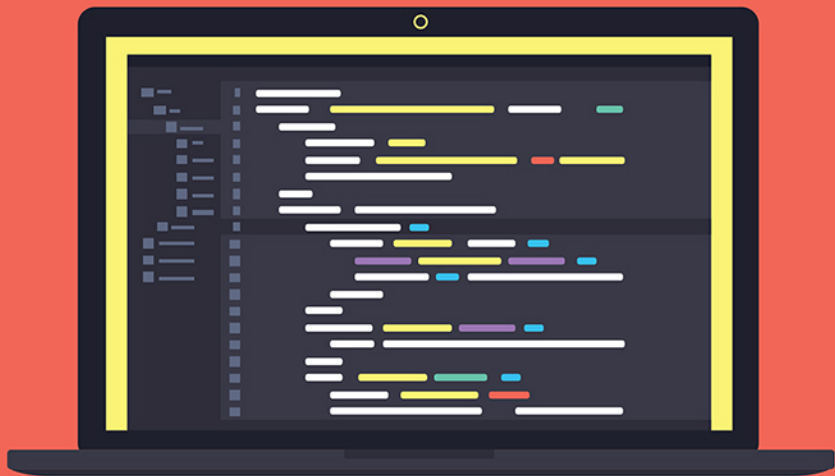


Java™ How to Program

Late Objects

ELEVENTH EDITION

Paul Deitel • Harvey Deitel



DIGITAL RESOURCES FOR STUDENTS

Your new textbook provides 12-month access to digital resources that may include VideoNotes (step-by-step video tutorials on programming concepts), source code, web chapters, quizzes, and more. Refer to the preface in the textbook for a detailed list of resources.

Follow the instructions below to register for the Companion Website for Paul Deitel and Harvey Deitel's *Java™ How to Program, Late Objects, Eleventh Edition, Global Edition*.

- 1 Go to **www.pearsonglobaleditions.com/deitel**.
- 2 Enter the title of your textbook or browse by author name.
- 3 Click Companion Website.
- 4 Click Register and follow the on-screen instructions to create a login name and password.

ISSLDO-WHIFF-SAURY-LAMBS-DOLBY-LIKES

Use the login name and password you created during registration to start using the digital resources that accompany your textbook.

IMPORTANT

This prepaid subscription does not include access to MyProgrammingLab, which is available at **www.myprogramminglab.com** for purchase.

This access code can only be used once. This subscription is valid for 12 months upon activation and is not transferable.

For technical support go to **<https://support.pearson.com/getsupport>**

Java™

HOW TO PROGRAM

LATE
OBJECTS

ELEVENTH EDITION
GLOBAL EDITION

Introducing
JShell

Use with
Java™ SE 8
or **Java™ SE 9**

Java™



HOW TO PROGRAM

LATE
OBJECTS

ELEVENTH EDITION
GLOBAL EDITION

Introducing
JShell

Paul Deitel

Deitel & Associates, Inc.

Harvey Deitel

Deitel & Associates, Inc.

Use with
Java™ SE 8
or **Java™ SE 9**



330 Hudson Street, NY, NY, 10013

Senior Vice President Courseware Portfolio Management: *Marcia J. Horton*
Director, Portfolio Management: Engineering, Computer Science & Global Editions: *Julian Partridge*
Higher Ed Portfolio Management: *Tracy Johnson (Dunkelberger)*
Portfolio Management Assistant: *Kristy Alaura*
Acquisitions Editor, Global Edition: *Aditee Agarwal*
Managing Content Producer: *Scott Disanno*
Content Producer: *Robert Engelhardt*
Senior Project Editor, Global Edition: *K.K. Neelakantan*
Web Developer: *Steve Wright*
Rights and Permissions Manager: *Ben Ferrini*
Manufacturing Buyer, Higher Ed, Lake Side Communications Inc (LSC): *Maura Zaldivar-Garcia*
Senior Manufacturing Controller, Global Edition: *Kay Holman*
Inventory Manager: *Ann Lam*
Product Marketing Manager: *Yvonne Vannatta*
Field Marketing Manager: *Demetrius Hall*
Marketing Assistant: *Jon Bryant*
Manager, Media Production, Global Edition: *Vikram Kumar*
Cover Designer: *Lumina Datamatics, Inc.*
Cover Art: ©*MchlSkbrv/Shutterstock*

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

Java™ and Netbeans™ screenshots ©2017 by Oracle Corporation, all rights reserved. Reprinted with permission.

Pearson Education Limited
KAO Two
KAO Park
Harlow
CM17 9SR
United Kingdom

and Associated Companies throughout the world

Visit us on the World Wide Web at: www.pearsonglobaleditions.com

© Pearson Education Limited 2020

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, Late Objects, 11th Edition, ISBN 978-0-13-479140-1 by Paul Deitel and Harvey Deitel published by Pearson Education © 2020.

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. For information regarding permissions, request forms, and the appropriate contacts within the Pearson Education Global Rights and Permissions department, please visit www.pearsoned.com/permissions.

This eBook is a standalone product and may or may not include all assets that were part of the print version. It also does not provide access to other Pearson digital products like MyLab and Mastering. The publisher reserves the right to remove any material in this eBook at any time.

British Library Cataloguing-in-Publication Data

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

ISBN 10: 1-292-27373-9
ISBN 13: 978-1-292-27373-0
eBook ISBN 13: 978-1-292-27374-7

eBook formatted by GEX Inc.

In memory of Dr. Henry Heimlich:

*Barbara Deitel used your Heimlich maneuver to
save Abbey Deitel's life. Our family is forever
grateful to you.*

Harvey, Barbara, Paul and Abbey Deitel

Trademarks

DEITEL and the double-thumbs-up bug 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® and Windows® 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 macOS 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

The online chapters and appendices listed at the end of this Table of Contents are located on the book's Companion Website (<http://www.pearsonglobaleditions.com>)—see the inside front cover of your book for details.

Foreword	25
Preface	27
Before You Begin	47
I Introduction to Computers, the Internet and Java	53
1.1 Introduction	54
1.2 Hardware and Software	56
1.2.1 Moore's Law	56
1.2.2 Computer Organization	57
1.3 Data Hierarchy	59
1.4 Machine Languages, Assembly Languages and High-Level Languages	61
1.5 Basic Introduction to Object Terminology	62
1.5.1 Automobile as an Object	63
1.5.2 Methods and Classes	63
1.5.3 Instantiation	63
1.5.4 Reuse	63
1.5.5 Messages and Method Calls	64
1.5.6 Attributes and Instance Variables	64
1.5.7 Encapsulation and Information Hiding	64
1.5.8 Inheritance	64
1.5.9 Interfaces	65
1.5.10 Object-Oriented Analysis and Design (OOAD)	65
1.5.11 The UML (Unified Modeling Language)	65
1.6 Operating Systems	66
1.6.1 Windows—A Proprietary Operating System	66
1.6.2 Linux—An Open-Source Operating System	66
1.6.3 Apple's macOS and Apple's iOS for iPhone®, iPad® and iPod Touch® Devices	67
1.6.4 Google's Android	67

8 Contents

1.7	Programming Languages	68
1.8	Java	70
1.9	A Typical Java Development Environment	71
1.10	Test-Driving a Java Application	74
1.11	Internet and World Wide Web	78
1.11.1	Internet: A Network of Networks	79
1.11.2	World Wide Web: Making the Internet User-Friendly	79
1.11.3	Web Services and Mashups	79
1.11.4	Internet of Things	80
1.12	Software Technologies	81
1.13	Getting Your Questions Answered	83

2 Introduction to Java Applications; Input/Output and Operators 87

2.1	Introduction	88
2.2	Your First Program in Java: Printing a Line of Text	88
2.2.1	Compiling the Application	92
2.2.2	Executing the Application	93
2.3	Modifying Your First Java Program	94
2.4	Displaying Text with <code>printf</code>	96
2.5	Another Application: Adding Integers	97
2.5.1	<code>import</code> Declarations	98
2.5.2	Declaring and Creating a Scanner to Obtain User Input from the Keyboard	98
2.5.3	Prompting the User for Input	99
2.5.4	Declaring a Variable to Store an Integer and Obtaining an Integer from the Keyboard	99
2.5.5	Obtaining a Second Integer	100
2.5.6	Using Variables in a Calculation	100
2.5.7	Displaying the Calculation Result	100
2.5.8	Java API Documentation	101
2.5.9	Declaring and Initializing Variables in Separate Statements	101
2.6	Memory Concepts	101
2.7	Arithmetic	102
2.8	Decision Making: Equality and Relational Operators	106
2.9	Wrap-Up	109

3 Control Statements: Part I; Assignment, ++ and -- Operators 120

3.1	Introduction	121
3.2	Algorithms	121
3.3	Pseudocode	122
3.4	Control Structures	122
3.4.1	Sequence Structure in Java	123

3.4.2	Selection Statements in Java	124
3.4.3	Iteration Statements in Java	124
3.4.4	Summary of Control Statements in Java	124
3.5	if Single-Selection Statement	125
3.6	if...else Double-Selection Statement	126
3.6.1	Nested if...else Statements	127
3.6.2	Dangling-else Problem	128
3.6.3	Blocks	128
3.6.4	Conditional Operator (?:)	129
3.7	while Iteration Statement	129
3.8	Formulating Algorithms: Counter-Controlled Iteration	131
3.9	Formulating Algorithms: Sentinel-Controlled Iteration	135
3.10	Formulating Algorithms: Nested Control Statements	142
3.11	Compound Assignment Operators	146
3.12	Increment and Decrement Operators	147
3.13	Primitive Types	150
3.14	Wrap-Up	150

4 Control Statements: Part 2; Logical Operators 164

4.1	Introduction	165
4.2	Essentials of Counter-Controlled Iteration	165
4.3	for Iteration Statement	166
4.4	Examples Using the for Statement	170
4.4.1	Application: Summing the Even Integers from 2 to 20	171
4.4.2	Application: Compound-Interest Calculations	172
4.5	do...while Iteration Statement	175
4.6	switch Multiple-Selection Statement	176
4.7	break and continue Statements	182
4.7.1	break Statement	182
4.7.2	continue Statement	182
4.8	Logical Operators	183
4.8.1	Conditional AND (&&) Operator	184
4.8.2	Conditional OR () Operator	184
4.8.3	Short-Circuit Evaluation of Complex Conditions	185
4.8.4	Boolean Logical AND (&) and Boolean Logical Inclusive OR () Operators	185
4.8.5	Boolean Logical Exclusive OR (^)	186
4.8.6	Logical Negation (!) Operator	186
4.8.7	Logical Operators Example	187
4.9	Structured-Programming Summary	189
4.10	Wrap-Up	194

5 Methods 204

5.1	Introduction	205
-----	--------------	-----

10 Contents

5.2	Program Units in Java	205
5.3	static Methods, static Variables and Class Math	207
5.4	Declaring Methods	209
5.5	Notes on Declaring and Using Methods	213
5.6	Method-Call Stack and Activation Records	214
5.6.1	Method-Call Stack	214
5.6.2	Stack Frames	214
5.6.3	Local Variables and Stack Frames	215
5.6.4	Stack Overflow	215
5.7	Argument Promotion and Casting	215
5.8	Java API Packages	216
5.9	Case Study: Secure Random-Number Generation	218
5.10	Case Study: A Game of Chance; Introducing enums	223
5.11	Scope of Declarations	227
5.12	Method Overloading	230
5.12.1	Declaring Overloaded Methods	230
5.12.2	Distinguishing Between Overloaded Methods	231
5.12.3	Return Types of Overloaded Methods	231
5.13	Wrap-Up	232

6 Arrays and ArrayLists 245

6.1	Introduction	246
6.2	Primitive Types vs. Reference Types	247
6.3	Arrays	247
6.4	Declaring and Creating Arrays	249
6.5	Examples Using Arrays	250
6.5.1	Creating and Initializing an Array	250
6.5.2	Using an Array Initializer	251
6.5.3	Calculating the Values to Store in an Array	252
6.5.4	Summing the Elements of an Array	253
6.5.5	Using Bar Charts to Display Array Data Graphically	254
6.5.6	Using the Elements of an Array as Counters	256
6.5.7	Using Arrays to Analyze Survey Results	257
6.6	Exception Handling: Processing the Incorrect Response	259
6.6.1	The try Statement	259
6.6.2	Executing the catch Block	259
6.6.3	toString Method of the Exception Parameter	260
6.7	Enhanced for Statement	260
6.8	Passing Arrays to Methods	261
6.9	Pass-By-Value vs. Pass-By-Reference	264
6.10	Multidimensional Arrays	264
6.10.1	Arrays of One-Dimensional Arrays	265
6.10.2	Two-Dimensional Arrays with Rows of Different Lengths	265
6.10.3	Creating Two-Dimensional Arrays with Array-Creation Expressions	266

6.10.4	Two-Dimensional Array Example: Displaying Element Values	266
6.10.5	Common Multidimensional-Array Manipulations Performed with <code>for</code> Statements	267
6.11	Variable-Length Argument Lists	268
6.12	Using Command-Line Arguments	269
6.13	Class Arrays	271
6.14	Introduction to Collections and Class <code>ArrayList</code>	274
6.15	Wrap-Up	278

7 Introduction to Classes and Objects 298

7.1	Introduction	299
7.2	Instance Variables, <i>set</i> Methods and <i>get</i> Methods	300
7.2.1	Account Class with an Instance Variable, and <i>set</i> and <i>get</i> Methods	300
7.2.2	AccountTest Class That Creates and Uses an Object of Class Account	302
7.2.3	Compiling and Executing an App with Multiple Classes	305
7.2.4	Account UML Class Diagram	305
7.2.5	Additional Notes on Class AccountTest	306
7.2.6	Software Engineering with <code>private</code> Instance Variables and <code>public set</code> and <i>get</i> Methods	307
7.3	Default and Explicit Initialization for Instance Variables	308
7.4	Account Class: Initializing Objects with Constructors	309
7.4.1	Declaring an Account Constructor for Custom Object Initialization	309
7.4.2	Class AccountTest: Initializing Account Objects When They're Created	310
7.5	Account Class with a Balance	312
7.5.1	Account Class with a <code>balance</code> Instance Variable of Type <code>double</code>	312
7.5.2	AccountTest Class to Use Class Account	313
7.6	Case Study: Card Shuffling and Dealing Simulation	316
7.7	Case Study: Class <code>GradeBook</code> Using an Array to Store Grades	320
7.8	Case Study: Class <code>GradeBook</code> Using a Two-Dimensional Array	326
7.9	Wrap-Up	331

8 Classes and Objects: A Deeper Look 339

8.1	Introduction	340
8.2	Time Class Case Study	340
8.3	Controlling Access to Members	345
8.4	Referring to the Current Object's Members with the <code>this</code> Reference	346
8.5	Time Class Case Study: Overloaded Constructors	348
8.6	Default and No-Argument Constructors	353
8.7	Notes on <i>Set</i> and <i>Get</i> Methods	354
8.8	Composition	355
8.9	<code>enum</code> Types	358

12 Contents

8.10	Garbage Collection	361
8.11	<code>static</code> Class Members	361
8.12	<code>static</code> Import	365
8.13	<code>final</code> Instance Variables	366
8.14	Package Access	367
8.15	Using <code>BigDecimal</code> for Precise Monetary Calculations	368
8.16	(Optional) GUI and Graphics Case Study: Using Objects with Graphics	371
8.17	Wrap-Up	375

9 Object-Oriented Programming: Inheritance 383

9.1	Introduction	384
9.2	Superclasses and Subclasses	385
9.3	<code>protected</code> Members	387
9.4	Relationship Between Superclasses and Subclasses	388
9.4.1	Creating and Using a <code>CommissionEmployee</code> Class	388
9.4.2	Creating and Using a <code>BasePlusCommissionEmployee</code> Class	393
9.4.3	Creating a <code>CommissionEmployee–BasePlusCommissionEmployee</code> Inheritance Hierarchy	398
9.4.4	<code>CommissionEmployee–BasePlusCommissionEmployee</code> Inheritance Hierarchy Using <code>protected</code> Instance Variables	401
9.4.5	<code>CommissionEmployee–BasePlusCommissionEmployee</code> Inheritance Hierarchy Using <code>private</code> Instance Variables	404
9.5	Constructors in Subclasses	408
9.6	Class Object	409
9.7	Designing with Composition vs. Inheritance	410
9.8	Wrap-Up	412

10 Object-Oriented Programming: Polymorphism and Interfaces 417

10.1	Introduction	418
10.2	Polymorphism Examples	420
10.3	Demonstrating Polymorphic Behavior	421
10.4	Abstract Classes and Methods	423
10.5	Case Study: Payroll System Using Polymorphism	426
10.5.1	Abstract Superclass <code>Employee</code>	427
10.5.2	Concrete Subclass <code>SalariedEmployee</code>	429
10.5.3	Concrete Subclass <code>HourlyEmployee</code>	431
10.5.4	Concrete Subclass <code>CommissionEmployee</code>	432
10.5.5	Indirect Concrete Subclass <code>BasePlusCommissionEmployee</code>	434
10.5.6	Polymorphic Processing, Operator <code>instanceof</code> and Downcasting	435
10.6	Allowed Assignments Between Superclass and Subclass Variables	440
10.7	<code>final</code> Methods and Classes	440
10.8	A Deeper Explanation of Issues with Calling Methods from Constructors	441
10.9	Creating and Using Interfaces	442
10.9.1	Developing a <code>Payable</code> Hierarchy	444

10.9.2	Interface Payable	445
10.9.3	Class Invoice	445
10.9.4	Modifying Class Employee to Implement Interface Payable	447
10.9.5	Using Interface Payable to Process Invoices and Employees Polymorphically	449
10.9.6	Some Common Interfaces of the Java API	450
10.10	Java SE 8 Interface Enhancements	451
10.10.1	default Interface Methods	451
10.10.2	static Interface Methods	452
10.10.3	Functional Interfaces	452
10.11	Java SE 9 private Interface Methods	453
10.12	private Constructors	453
10.13	Program to an Interface, Not an Implementation	454
10.13.1	Implementation Inheritance Is Best for Small Numbers of Tightly Coupled Classes	454
10.13.2	Interface Inheritance Is Best for Flexibility	454
10.13.3	Rethinking the Employee Hierarchy	455
10.14	(Optional) GUI and Graphics Case Study: Drawing with Polymorphism	456
10.15	Wrap-Up	458

11 Exception Handling: A Deeper Look **465**

11.1	Introduction	466
11.2	Example: Divide by Zero without Exception Handling	467
11.3	Example: Handling ArithmeticExceptions and InputMismatchExceptions	469
11.4	When to Use Exception Handling	475
11.5	Java Exception Hierarchy	475
11.6	finally Block	479
11.7	Stack Unwinding and Obtaining Information from an Exception	483
11.8	Chained Exceptions	486
11.9	Declaring New Exception Types	488
11.10	Preconditions and Postconditions	489
11.11	Assertions	489
11.12	try-with-Resources: Automatic Resource Deallocation	491
11.13	Wrap-Up	492

12 JavaFX Graphical User Interfaces: Part I **498**

12.1	Introduction	499
12.2	JavaFX Scene Builder	500
12.3	JavaFX App Window Structure	501
12.4	>Welcome App—Displaying Text and an Image	502
12.4.1	Opening Scene Builder and Creating the File Welcome.fxml	502
12.4.2	Adding an Image to the Folder Containing Welcome.fxml	503
12.4.3	Creating a VBox Layout Container	503
12.4.4	Configuring the VBox Layout Container	504
12.4.5	Adding and Configuring a Label	504

14 Contents

12.4.6	Adding and Configuring an ImageView	505
12.4.7	Previewing the Welcome GUI	507
12.5	Tip Calculator App —Introduction to Event Handling	507
12.5.1	Test-Driving the Tip Calculator App	508
12.5.2	Technologies Overview	509
12.5.3	Building the App's GUI	511
12.5.4	TipCalculator Class	518
12.5.5	TipCalculatorController Class	520
12.6	Features Covered in the Other JavaFX Chapters	525
12.7	Wrap-Up	525

13 JavaFX GUI: Part 2 **533**

13.1	Introduction	534
13.2	Laying Out Nodes in a Scene Graph	534
13.3	Painter App: RadioButtons, Mouse Events and Shapes	536
13.3.1	Technologies Overview	536
13.3.2	Creating the Painter.fxml File	538
13.3.3	Building the GUI	538
13.3.4	Painter Subclass of Application	541
13.3.5	PainterController Class	542
13.4	Color Chooser App: Property Bindings and Property Listeners	546
13.4.1	Technologies Overview	546
13.4.2	Building the GUI	547
13.4.3	ColorChooser Subclass of Application	549
13.4.4	ColorChooserController Class	550
13.5	Cover Viewer App: Data-Driven GUIs with JavaFX Collections	552
13.5.1	Technologies Overview	553
13.5.2	Adding Images to the App's Folder	553
13.5.3	Building the GUI	553
13.5.4	CoverViewer Subclass of Application	555
13.5.5	CoverViewerController Class	555
13.6	Cover Viewer App: Customizing ListView Cells	557
13.6.1	Technologies Overview	558
13.6.2	Copying the CoverViewer App	558
13.6.3	ImageTextCell Custom Cell Factory Class	559
13.6.4	CoverViewerController Class	560
13.7	Additional JavaFX Capabilities	561
13.8	JavaFX 9: Java SE 9 JavaFX Updates	563
13.9	Wrap-Up	565

14 Strings, Characters and Regular Expressions **574**

14.1	Introduction	575
14.2	Fundamentals of Characters and Strings	575
14.3	Class String	576
14.3.1	String Constructors	576

14.3.2	String Methods <code>length</code> , <code>charAt</code> and <code>getChars</code>	577
14.3.3	Comparing Strings	579
14.3.4	Locating Characters and Substrings in Strings	583
14.3.5	Extracting Substrings from Strings	585
14.3.6	Concatenating Strings	586
14.3.7	Miscellaneous String Methods	587
14.3.8	String Method <code>valueOf</code>	588
14.4	Class <code>StringBuilder</code>	589
14.4.1	<code>StringBuilder</code> Constructors	590
14.4.2	<code>StringBuilder</code> Methods <code>length</code> , <code>capacity</code> , <code>setLength</code> and <code>ensureCapacity</code>	591
14.4.3	<code>StringBuilder</code> Methods <code>charAt</code> , <code>setCharAt</code> , <code>getChars</code> and <code>reverse</code>	592
14.4.4	<code>StringBuilder</code> <code>append</code> Methods	593
14.4.5	<code>StringBuilder</code> Insertion and Deletion Methods	595
14.5	Class <code>Character</code>	596
14.6	Tokenizing Strings	601
14.7	Regular Expressions, Class <code>Pattern</code> and Class <code>Matcher</code>	602
14.7.1	Replacing Substrings and Splitting Strings	607
14.7.2	Classes <code>Pattern</code> and <code>Matcher</code>	609
14.8	Wrap-Up	611

15 Files, Input/Output Streams, NIO and XML Serialization **622**

15.1	Introduction	623
15.2	Files and Streams	623
15.3	Using NIO Classes and Interfaces to Get File and Directory Information	625
15.4	Sequential Text Files	629
15.4.1	Creating a Sequential Text File	629
15.4.2	Reading Data from a Sequential Text File	632
15.4.3	Case Study: A Credit-Inquiry Program	633
15.4.4	Updating Sequential Files	638
15.5	XML Serialization	638
15.5.1	Creating a Sequential File Using XML Serialization	638
15.5.2	Reading and Deserializing Data from a Sequential File	644
15.6	<code>FileChooser</code> and <code>DirectoryChooser</code> Dialogs	645
15.7	(Optional) Additional <code>java.io</code> Classes	651
15.7.1	Interfaces and Classes for Byte-Based Input and Output	651
15.7.2	Interfaces and Classes for Character-Based Input and Output	653
15.8	Wrap-Up	654

16 Generic Collections **662**

16.1	Introduction	663
16.2	Collections Overview	663

16 Contents

16.3	Type-Wrapper Classes	665
16.4	Autoboxing and Auto-Unboxing	665
16.5	Interface Collection and Class Collections	665
16.6	Lists	666
16.6.1	ArrayList and Iterator	667
16.6.2	LinkedList	669
16.7	Collections Methods	674
16.7.1	Method sort	674
16.7.2	Method shuffle	678
16.7.3	Methods reverse, fill, copy, max and min	680
16.7.4	Method binarySearch	682
16.7.5	Methods addAll, frequency and disjoint	683
16.8	Class PriorityQueue and Interface Queue	685
16.9	Sets	686
16.10	Maps	689
16.11	Synchronized Collections	693
16.12	Unmodifiable Collections	693
16.13	Abstract Implementations	694
16.14	Java SE 9: Convenience Factory Methods for Immutable Collections	694
16.15	Wrap-Up	698

17 Lambdas and Streams

704

17.1	Introduction	705
17.2	Streams and Reduction	707
17.2.1	Summing the Integers from 1 through 10 with a for Loop	707
17.2.2	External Iteration with for Is Error Prone	708
17.2.3	Summing with a Stream and Reduction	708
17.2.4	Internal Iteration	709
17.3	Mapping and Lambdas	710
17.3.1	Lambda Expressions	711
17.3.2	Lambda Syntax	712
17.3.3	Intermediate and Terminal Operations	713
17.4	Filtering	714
17.5	How Elements Move Through Stream Pipelines	716
17.6	Method References	717
17.6.1	Creating an IntStream of Random Values	718
17.6.2	Performing a Task on Each Stream Element with forEach and a Method Reference	718
17.6.3	Mapping Integers to String Objects with mapToObj	719
17.6.4	Concatenating Strings with collect	719
17.7	IntStream Operations	720
17.7.1	Creating an IntStream and Displaying Its Values	721
17.7.2	Terminal Operations count, min, max, sum and average	721
17.7.3	Terminal Operation reduce	722
17.7.4	Sorting IntStream Values	724

17.8	Functional Interfaces	725
17.9	Lambdas: A Deeper Look	726
17.10	Stream<Integer> Manipulations	727
17.10.1	Creating a Stream<Integer>	728
17.10.2	Sorting a Stream and Collecting the Results	729
17.10.3	Filtering a Stream and Storing the Results for Later Use	729
17.10.4	Filtering and Sorting a Stream and Collecting the Results	730
17.10.5	Sorting Previously Collected Results	730
17.11	Stream<String> Manipulations	730
17.11.1	Mapping Strings to Uppercase	731
17.11.2	Filtering Strings Then Sorting Them in Case-Insensitive Ascending Order	732
17.11.3	Filtering Strings Then Sorting Them in Case-Insensitive Descending Order	732
17.12	Stream<Employee> Manipulations	733
17.12.1	Creating and Displaying a List<Employee>	734
17.12.2	Filtering Employees with Salaries in a Specified Range	735
17.12.3	Sorting Employees By Multiple Fields	738
17.12.4	Mapping Employees to Unique-Last-Name Strings	740
17.12.5	Grouping Employees By Department	741
17.12.6	Counting the Number of Employees in Each Department	742
17.12.7	Summing and Averaging Employee Salaries	743
17.13	Creating a Stream<String> from a File	744
17.14	Streams of Random Values	747
17.15	Infinite Streams	749
17.16	Lambda Event Handlers	751
17.17	Additional Notes on Java SE 8 Interfaces	751
17.18	Wrap-Up	752

18 Recursion 766

18.1	Introduction	767
18.2	Recursion Concepts	768
18.3	Example Using Recursion: Factorials	769
18.4	Reimplementing Class FactorialCalculator Using BigInteger	771
18.5	Example Using Recursion: Fibonacci Series	773
18.6	Recursion and the Method-Call Stack	776
18.7	Recursion vs. Iteration	777
18.8	Towers of Hanoi	779
18.9	Fractals	781
18.9.1	Koch Curve Fractal	782
18.9.2	(Optional) Case Study: Lo Feather Fractal	783
18.9.3	(Optional) Fractal App GUI	785
18.9.4	(Optional) FractalController Class	787
18.10	Recursive Backtracking	792
18.11	Wrap-Up	792

19	Searching, Sorting and Big O	801
19.1	Introduction	802
19.2	Linear Search	803
19.3	Big O Notation	806
19.3.1	$O(1)$ Algorithms	806
19.3.2	$O(n)$ Algorithms	806
19.3.3	$O(n^2)$ Algorithms	806
19.3.4	Big O of the Linear Search	807
19.4	Binary Search	807
19.4.1	Binary Search Implementation	808
19.4.2	Efficiency of the Binary Search	811
19.5	Sorting Algorithms	812
19.6	Selection Sort	812
19.6.1	Selection Sort Implementation	813
19.6.2	Efficiency of the Selection Sort	815
19.7	Insertion Sort	815
19.7.1	Insertion Sort Implementation	816
19.7.2	Efficiency of the Insertion Sort	818
19.8	Merge Sort	819
19.8.1	Merge Sort Implementation	819
19.8.2	Efficiency of the Merge Sort	824
19.9	Big O Summary for This Chapter's Searching and Sorting Algorithms	824
19.10	Massive Parallelism and Parallel Algorithms	825
19.11	Wrap-Up	825
20	Generic Classes and Methods: A Deeper Look	831
20.1	Introduction	832
20.2	Motivation for Generic Methods	832
20.3	Generic Methods: Implementation and Compile-Time Translation	834
20.4	Additional Compile-Time Translation Issues: Methods That Use a Type Parameter as the Return Type	837
20.5	Overloading Generic Methods	840
20.6	Generic Classes	841
20.7	Wildcards in Methods That Accept Type Parameters	848
20.8	Wrap-Up	852
21	Custom Generic Data Structures	856
21.1	Introduction	857
21.2	Self-Referential Classes	858
21.3	Dynamic Memory Allocation	858
21.4	Linked Lists	859
21.4.1	Singly Linked Lists	859
21.4.2	Implementing a Generic List Class	860
21.4.3	Generic Classes ListNode and List	863

21.4.4	Class <code>ListTest</code>	863
21.4.5	List Method <code>insertAtFront</code>	865
21.4.6	List Method <code>insertAtBack</code>	866
21.4.7	List Method <code>removeFromFront</code>	866
21.4.8	List Method <code>removeFromBack</code>	867
21.4.9	List Method <code>print</code>	868
21.4.10	Creating Your Own Packages	868
21.5	Stacks	873
21.6	Queues	876
21.7	Trees	878
21.8	Wrap-Up	885

22 JavaFX Graphics and Multimedia 910

22.1	Introduction	911
22.2	Controlling Fonts with Cascading Style Sheets (CSS)	912
22.2.1	CSS That Styles the GUI	912
22.2.2	FXML That Defines the GUI—Introduction to XML Markup	915
22.2.3	Referencing the CSS File from FXML	918
22.2.4	Specifying the <code>VBox</code> 's Style Class	918
22.2.5	Programmatically Loading CSS	918
22.3	Displaying Two-Dimensional Shapes	919
22.3.1	Defining Two-Dimensional Shapes with FXML	919
22.3.2	CSS That Styles the Two-Dimensional Shapes	922
22.4	<code>PolyLines</code> , <code>Polygons</code> and <code>Paths</code>	924
22.4.1	GUI and CSS	925
22.4.2	<code>PolyShapesController</code> Class	926
22.5	Transforms	929
22.6	Playing Video with <code>Media</code> , <code>MediaPlayer</code> and <code>MediaViewer</code>	931
22.6.1	<code>VideoPlayer</code> GUI	932
22.6.2	<code>VideoPlayerController</code> Class	934
22.7	Transition Animations	938
22.7.1	<code>TransitionAnimations.fxml</code>	938
22.7.2	<code>TransitionAnimationsController</code> Class	940
22.8	Timeline Animations	944
22.9	Frame-by-Frame Animation with <code>AnimationTimer</code>	947
22.10	Drawing on a Canvas	949
22.11	Three-Dimensional Shapes	954
22.12	Wrap-Up	957

23 Concurrency 973

23.1	Introduction	974
23.2	Thread States and Life Cycle	976
23.2.1	<i>New</i> and <i>Runnable</i> States	977
23.2.2	<i>Waiting</i> State	977

23.2.3	<i>Timed Waiting State</i>	977
23.2.4	<i>Blocked State</i>	977
23.2.5	<i>Terminated State</i>	977
23.2.6	Operating-System View of the <i>Runnable State</i>	978
23.2.7	Thread Priorities and Thread Scheduling	978
23.2.8	Indefinite Postponement and Deadlock	979
23.3	Creating and Executing Threads with the Executor Framework	979
23.4	Thread Synchronization	983
23.4.1	Immutable Data	984
23.4.2	Monitors	984
23.4.3	Unsynchronized Mutable Data Sharing	985
23.4.4	Synchronized Mutable Data Sharing—Making Operations Atomic	989
23.5	Producer/Consumer Relationship without Synchronization	992
23.6	Producer/Consumer Relationship: <code>ArrayBlockingQueue</code>	1000
23.7	(Advanced) Producer/Consumer Relationship with <code>synchronized</code> , <code>wait</code> , <code>notify</code> and <code>notifyAll</code>	1003
23.8	(Advanced) Producer/Consumer Relationship: Bounded Buffers	1009
23.9	(Advanced) Producer/Consumer Relationship: The Lock and Condition Interfaces	1017
23.10	Concurrent Collections	1024
23.11	Multithreading in JavaFX	1026
23.11.1	Performing Computations in a Worker Thread: Fibonacci Numbers	1027
23.11.2	Processing Intermediate Results: Sieve of Eratosthenes	1032
23.12	<code>sort/parallelSort</code> Timings with the Java SE 8 Date/Time API	1038
23.13	Java SE 8: Sequential vs. Parallel Streams	1041
23.14	(Advanced) Interfaces <code>Callable</code> and <code>Future</code>	1043
23.15	(Advanced) Fork/Join Framework	1048
23.16	Wrap-Up	1048

24 Accessing Databases with JDBC 1060

24.1	Introduction	1061
24.2	Relational Databases	1062
24.3	A books Database	1063
24.4	SQL	1067
24.4.1	Basic SELECT Query	1068
24.4.2	WHERE Clause	1068
24.4.3	ORDER BY Clause	1070
24.4.4	Merging Data from Multiple Tables: INNER JOIN	1072
24.4.5	INSERT Statement	1073
24.4.6	UPDATE Statement	1074
24.4.7	DELETE Statement	1075
24.5	Setting Up a Java DB Database	1076
24.5.1	Creating the Chapter's Databases on Windows	1077

24.5.2	Creating the Chapter's Databases on macOS	1078
24.5.3	Creating the Chapter's Databases on Linux	1078
24.6	Connecting to and Querying a Database	1078
24.6.1	Automatic Driver Discovery	1080
24.6.2	Connecting to the Database	1080
24.6.3	Creating a Statement for Executing Queries	1081
24.6.4	Executing a Query	1081
24.6.5	Processing a Query's ResultSet	1082
24.7	Querying the books Database	1083
24.7.1	ResultSetTableModel Class	1083
24.7.2	DisplayQueryResults App's GUI	1090
24.7.3	DisplayQueryResultsController Class	1090
24.8	RowSet Interface	1095
24.9	PreparedStatement	1098
24.9.1	AddressBook App That Uses PreparedStatement	1099
24.9.2	Class Person	1099
24.9.3	Class PersonQueries	1101
24.9.4	AddressBook GUI	1104
24.9.5	Class AddressBookController	1105
24.10	Stored Procedures	1110
24.11	Transaction Processing	1110
24.12	Wrap-Up	1111

25 Introduction to JShell: Java 9's REPL for Interactive Java

1119

25.1	Introduction	1120
25.2	Installing JDK 9	1122
25.3	Introduction to JShell	1122
25.3.1	Starting a JShell Session	1123
25.3.2	Executing Statements	1123
25.3.3	Declaring Variables Explicitly	1124
25.3.4	Listing and Executing Prior Snippets	1126
25.3.5	Evaluating Expressions and Declaring Variables Implicitly	1128
25.3.6	Using Implicitly Declared Variables	1128
25.3.7	Viewing a Variable's Value	1129
25.3.8	Resetting a JShell Session	1129
25.3.9	Writing Multiline Statements	1129
25.3.10	Editing Code Snippets	1130
25.3.11	Exiting JShell	1133
25.4	Command-Line Input in JShell	1133
25.5	Declaring and Using Classes	1134
25.5.1	Creating a Class in JShell	1135
25.5.2	Explicitly Declaring Reference-Type Variables	1135
25.5.3	Creating Objects	1136
25.5.4	Manipulating Objects	1136

22 Contents

25.5.5	Creating a Meaningful Variable Name for an Expression	1137
25.5.6	Saving and Opening Code-Snippet Files	1138
25.6	Discovery with JShell Auto-Completion	1138
25.6.1	Auto-Completing Identifiers	1139
25.6.2	Auto-Completing JShell Commands	1140
25.7	Exploring a Class's Members and Viewing Documentation	1140
25.7.1	Listing Class <code>Math</code> 's <code>static</code> Members	1141
25.7.2	Viewing a Method's Parameters	1141
25.7.3	Viewing a Method's Documentation	1142
25.7.4	Viewing a <code>public</code> Field's Documentation	1142
25.7.5	Viewing a Class's Documentation	1143
25.7.6	Viewing Method Overloads	1143
25.7.7	Exploring Members of a Specific Object	1144
25.8	Declaring Methods	1146
25.8.1	Forward Referencing an Undeclared Method—Declaring Method <code>displayCubes</code>	1146
25.8.2	Declaring a Previously Undeclared Method	1146
25.8.3	Testing <code>cube</code> and Replacing Its Declaration	1147
25.8.4	Testing Updated Method <code>cube</code> and Method <code>displayCubes</code>	1147
25.9	Exceptions	1148
25.10	Importing Classes and Adding Packages to the <code>CLASSPATH</code>	1149
25.11	Using an External Editor	1151
25.12	Summary of JShell Commands	1153
25.12.1	Getting Help in JShell	1154
25.12.2	<code>/edit</code> Command: Additional Features	1155
25.12.3	<code>/reload</code> Command	1155
25.12.4	<code>/drop</code> Command	1156
25.12.5	Feedback Modes	1156
25.12.6	Other JShell Features Configurable with <code>/set</code>	1158
25.13	Keyboard Shortcuts for Snippet Editing	1159
25.14	How JShell Reinterprets Java for Interactive Use	1159
25.15	IDE JShell Support	1160
25.16	Wrap-Up	1160

Chapters on the Web 1176

A Operator Precedence Chart 1177

B ASCII Character Set 1179

C Keywords and Reserved Words 1180

D Primitive Types 1181

E	Using the Debugger	I 182
E.1	Introduction	1183
E.2	Breakpoints and the run, stop, cont and print Commands	1183
E.3	The print and set Commands	1187
E.4	Controlling Execution Using the step, step up and next Commands	1189
E.5	The watch Command	1191
E.6	The clear Command	1193
E.7	Wrap-Up	1196
	Appendices on the Web	I 197
	Index	I 199

Online Chapters and Appendices

The online chapters and appendices are located on the book's Companion Website. See the book's inside front cover for details.

- 26** Swing GUI Components: Part 1
- 27** Graphics and Java 2D
- 28** Networking
- 29** Java Persistence API (JPA)
- 30** JavaServer™ Faces Web Apps: Part 1
- 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
- 35** Swing GUI Components: Part 2
- 36** Java Module System and Other Java 9 Features

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

Throughout my career I've met and interviewed many expert Java developers who've learned from Paul and Harvey, through one or more of their college textbooks, professional books, videos and corporate training. Many Java User Groups have joined together around the Deitels' publications, which are used internationally in university courses and professional training programs. You are joining an elite group.

How do I become an expert Java developer?

This is one of the most common questions I receive at talks for university students and at events with Java professionals. Students want to become expert developers—and this is a great time to be one.

The market is wide open, full of opportunities and fascinating projects, especially for those who take the time to learn, practice and master software development. The world needs good, focused expert developers.

So, how do you do it? First, let's be clear: Software development is hard. But do not be discouraged. Mastering it opens the door to great opportunities. Accept that it's hard, embrace the complexity, enjoy the ride. There are no limits to how much you can expand your skills.

Software development is an amazing skill. It can take you anywhere. You can work in any field. From nonprofits making the world a better place, to bleeding-edge biological technologies. From the frenetic daily run of the financial world to the deep mysteries of religion. From sports to music to acting. Everything has software. The success or failure of initiatives everywhere will depend on developers' knowledge and skills.

The push for you to get the relevant skills is what makes *Java How to Program, 11/e* so compelling. Written for students and new developers, it's easy to follow. It's written by authors who are educators and developers, with input over the years from some of the world's leading academics and professional Java experts—Java Champions, open-source Java developers, even creators of Java itself. Their collective knowledge and experience will guide you. Even seasoned Java professionals will learn and grow their expertise with the wisdom in these pages.

How can this book help you become an expert?

Java was released in 1995—Paul and Harvey had the first edition of *Java How to Program* ready for Fall 1996 classes. Since that groundbreaking book, they've produced ten more editions, keeping current with the latest developments and idioms in the Java software-engineering community. You hold in your hands the map that will enable you to rapidly develop your Java skills.

The Deitels have broken down the humongous Java world into well-defined, specific goals. Put in your full attention, and consciously “beat” each chapter. You'll soon find

yourself moving nicely along your road to excellence. And with both Java 8 and Java 9 in the same book, you'll have up-to-date skills on the latest Java technologies.

Most importantly, this book is not just meant for you to read—it's meant for you to practice. Be it in the classroom or at home after work, experiment with the abundant sample code and practice with the book's extraordinarily rich and diverse collection of exercises. Take the time to do all that is in here and you'll be well on your way to achieving a level of expertise that will challenge professional developers out there. After working with Java for more than 20 years, I can tell you that this is not an exaggeration.

For example, one of my favorite chapters is Lambdas and Streams. The chapter covers the topic in detail and the exercises shine—many real-world challenges that developers will encounter every day and that will help you sharpen your skills. After solving these exercises, novices and experienced developers alike will deeply understand these important Java features. And if you have a question, don't be shy—the Deitels publish their email address in every book they write to encourage interaction.

That's also why I love the chapter about JShell—the new Java 9 tool that enables interactive Java. JShell allows you to explore, discover and experiment with new concepts, language features and APIs, make mistakes—accidentally and intentionally—and correct them, and rapidly prototype new code. It may prove to be the most important tool for leveraging your learning and productivity. Paul and Harvey give a full treatment of JShell that both students and experienced developers will be able to put to use immediately.

I'm impressed with the care that the Deitels always take care to accommodate readers at all levels. They ease you into difficult concepts and deal with the challenges that professionals will encounter in industry projects.

There's lots of information about Java 9, the important new Java release. You can jump right in and learn the latest Java features. If you're still working with Java 8, you can ease into Java 9 at your own pace—be sure to begin with the extraordinary JShell coverage.

Another example is the amazing coverage of JavaFX—Java's latest GUI, graphics and multimedia capabilities. JavaFX is the recommended toolkit for new projects. But if you'll be working on legacy projects that use the older Swing API, those chapters are still available to you.

Make sure to dig in on Paul and Harvey's treatment of concurrency. They explain the basic concepts so clearly that the intermediate and advanced examples and discussions will be easy to master. You will be ready to maximize your applications' performance in an increasingly multi-core world.

I encourage you to participate in the worldwide Java community. There are many helpful folks out there who stand ready to help you. Ask questions, get answers and answer your peers' questions. Along with this book, the Internet and the academic and professional communities will help speed you on your way to becoming an expert Java developer. I wish you success!

Bruno Sousa
bruno@javaman.com.br
Java Champion
Java Specialist at ToolsCloud
President of SouJava (the Brazilian Java Society)
SouJava representative at the Java Community Process



Preface

Welcome to the Java programming language and *Java How to Program, Late Objects, Eleventh 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 professional societies,¹ and for *Advanced Placement (AP) Computer Science* exam preparation.² It also will help you prepare for most topics covered by the following Oracle Java Standard Edition 8 (Java SE 8) Certifications:³

- Oracle Certified Associate, Java SE 8 Programmer
- Oracle Certified Professional, Java SE 8 Programmer

If you haven't already done so, please read the bullet points and reviewer comments on the back cover and inside back cover—these concisely capture the essence of the book. In this Preface we provide more detail for students, instructors and professionals.

Our primary goal is to prepare college students to meet the Java programming challenges they'll encounter in upper-level courses and in industry. We focus on software engineering best practices. At the heart of the book is the Deitel signature **live-code approach**—we present most concepts in the context of hundreds of complete working programs that have been tested on **Windows**[®], **macOS**[®] and **Linux**[®]. The complete code examples are accompanied by live sample executions.

New and Updated Features

In the following sections, we discuss the key features and updates we've made for *Java How to Program, 11/e*, including:

- Flexibility Using **Java SE 8 or the New Java SE 9** (which includes Java SE 8)
- *Java How to Program, 11/e's Modular Organization*
- Introduction and Programming Fundamentals
- Flexible Coverage of **Java SE 9: JShell, the Module System** and Other Java SE 9 Topics
- **Object-Oriented Programming**
- Flexible **JavaFX/Swing** GUI, Graphics, Animation and Video Coverage

-
1. *Computer Science Curricula 2013 Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*, December 20, 2013, The Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM), IEEE Computer Society.
 2. <https://apstudent.collegeboard.org/apcourse/ap-computer-science-a/exam-practice>
 3. <http://bit.ly/OracleJavaSE8Certification> (At the time of this writing, the Java SE 9 certification exams were not yet available.)

- Data Structures and than
- Flexible Lambdas and Streams Coverage
- Concurrency and Multi-Core Performance
- Database: JDBC and JPA
- Web-Application Development and Web Services
- Optional Online Object-Oriented Design Case Study

Flexibility Using Java SE 8 or the New Java SE 9

8
9

To meet the needs of our diverse audiences, we designed the book for college and professional courses based on **Java SE 8 or Java SE 9**, which from this point forward we'll refer to simply as Java 8 and Java 9, respectively. Each feature first introduced in Java 8 or Java 9 is accompanied by an 8 or 9 icon in the margin, like those to the left of this paragraph. The new Java 9 capabilities are covered in clearly marked, *easy-to-include-or-omit* chapters and sections—some in the print book and some online. Figures 1 and 2 list some key Java 8 and Java 9 features that we cover, respectively.

Java 8 features

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

Fig. 1 | Some key features we cover that were introduced in Java 8.

Java 9 features

<i>In the Print Book</i>	<i>On the Companion Website</i>
New JShell chapter	Module system
<code>_</code> is no longer allowed as an identifier	HTML5 Javadoc enhancements
private interface methods	Matcher class's new method overloads
Effectively final variables can be used in try- with-resources statements	CompletableFuture enhancements
Mention of the Stack Walking API	JavaFX 9 skin APIs and other enhancements
Mention of JEP 254, Compact Strings	Mentions of:
Collection factory methods	Overview of Java 9 security enhancements
	G1 garbage collector
	Object serialization security enhancements
	Enhanced deprecation

Fig. 2 | Some key new features we cover that were introduced in Java 9.

Java How to Program, Late Objects, 11/e's Modular Organization

The book's modular organization helps instructors plan their syllabi.

Java How to Program, Late Objects, 11/e, is appropriate for programming courses at various levels. Chapters 1–25 are popular in core CS 1 and CS 2 courses and introductory course sequences in related disciplines—these chapters appear in the **print book**. Chapters 26–36 are intended for advanced courses and are located on the book's **Companion Website**.

Part 1: Introduction

Chapter 1, Introduction to Computers, the Internet and Java

Chapter 2, Introduction to Java Applications; Input/Output and Operators

Chapter 25, Introduction to JShell: Java 9's REPL for Interactive Java

Part 2: Additional Programming Fundamentals

Chapter 3, Control Statements: Part 1; Assignment, ++ and -- Operators

Chapter 4, Control Statements: Part 2; Logical Operators

Chapter 5, Methods

Chapter 6, Arrays and ArrayLists

Chapter 14, Strings, Characters and Regular Expressions

Chapter 15, Files, Input/Output Streams, NIO and XML Serialization

Part 3: Object-Oriented Programming

Chapter 7, Introduction to Classes and Objects

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

Part 4: JavaFX Graphical User Interfaces, Graphics and Multimedia

Chapter 12, JavaFX Graphical User Interfaces: Part 1

Chapter 13, JavaFX GUI: Part 2

Chapter 22, JavaFX Graphics and Multimedia

Part 5: Data Structures, Generic Collections, Lambdas and Streams

Chapter 16, Generic Collections

Chapter 17, Lambdas and Streams

Chapter 18, Recursion

Chapter 19, Searching, Sorting and Big O

Chapter 20, Generic Classes and Methods: A Deeper Look

Chapter 21, Custom Generic Data Structures

Part 6: Concurrency; Networking

Chapter 23, Concurrency

Chapter 28, Networking

Part 7: Database-Driven Desktop Development

Chapter 24, Accessing Databases with JDBC

Chapter 29, Java Persistence API (JPA)

Part 8: Web App Development and Web Services

Chapter 30, JavaServer™ Faces Web Apps: Part 1

Chapter 31, JavaServer™ Faces Web Apps: Part 2

Chapter 32, REST Web Services

Part 9: Other Java 9 Topics

Chapter 36, Java Module System and Other Java 9 Features

Part 10: (Optional) Object-Oriented Design

Chapter 33, ATM Case Study, Part 1: Object-Oriented Design with the UML

Chapter 34, ATM Case Study Part 2: Implementing an Object-Oriented Design

Part 11: (Optional) Swing Graphical User Interfaces and Java 2D Graphics

Chapter 26, Swing GUI Components: Part 1

Chapter 27, Graphics and Java 2D

Chapter 35, Swing GUI Components: Part 2

Introduction and Programming Fundamentals (Parts 1 and 2)

Chapters 1 through 7 provide a friendly, example-driven treatment of traditional introductory programming topics. This book features a **late objects approach**—see the section “Object-Oriented Programming” later in this Preface. Note in the preceding outline that Part 1 includes the (optional) Chapter 25 on Java 9’s new JShell. Instructors and students who cover JShell will appreciate how its interactivity makes Java “come alive,” leveraging the learning process—see the next section.

9 Flexible Coverage of Java 9: JShell, the Module System and Other Java 9 Topics

JShell: Java 9’s REPL (Read-Eval-Print-Loop) for Interactive Java

JShell provides a friendly environment that enables you to quickly explore, discover and experiment with Java’s language features and its extensive libraries. JShell replaces the tedious cycle of editing, compiling and executing with its **read-evaluate-print-loop**. Rather than complete programs, you write JShell commands and Java code snippets. When you enter a snippet, JShell *immediately*

- reads it,
- evaluates it and

- **prints** messages that help you see the effects of your code, then it
- **loops** to perform this process again for the next snippet.

As you work through Chapter 25’s scores of examples and exercises, you’ll see how JShell and its **instant feedback** keep your attention, enhance your performance and speed the learning and software development processes.

As a student you’ll find JShell easy and fun to use. It will help you learn Java features faster and more deeply and will help you verify that these features work the way they’re supposed to. As an instructor, you’ll appreciate how JShell encourages your students to dig in, and that it **leverages the learning process**. As a professional you’ll appreciate how JShell helps you rapidly prototype key code segments and how it helps you discover and experiment with new APIs. **If you’re staying with Java 8 for a while, you can install JDKs 8 and 9 side-by-side and use JShell on JDK 9 for experimentation. The JDK 9 section of the Before You Begin that follows this Preface shows how to manage multiple JDKs on Windows, macOS and Linux.**

The JShell content is packaged modularly in Chapter 25. The chapter:

1. is **easy to include or omit**.
2. is organized as a series of 16 sections, many of which are designed to be covered after a specific earlier chapter of the book (Fig. 3).
3. offers rich coverage of JShell’s capabilities. It’s **example-intensive**—you should do each of the examples. Get JShell into your fingertips. You’ll appreciate how quickly and conveniently you can do things.
4. includes **dozens of Self-Review Exercises, each with an answer**. These exercises can be done after you read **Chapter 2** and **Section 25.3**. As you do each of them, flip the page and check your answer. This will help you master the basics of JShell quickly. Then as you do each of the **examples** in the remainder of the chapter you’ll master the vast majority of JShell’s capabilities.

JShell discussions	Can be covered after
Section 25.3 introduces JShell , including starting a session, executing statements, declaring variables, evaluating expressions, JShell’s type-inference capabilities and more.	Chapter 2, Introduction to Java Applications; Input/Output and Operators
Section 25.4 discusses command-line input with Scanner in JShell.	
Section 25.5 discusses how to declare and use classes in JShell, including how to load a Java source-code file containing an existing class declaration.	Chapter 7, Introduction to Classes and Objects
Section 25.6 shows how to use JShell’s auto-completion capabilities to discover a class’s capabilities and JShell commands.	

Fig. 3 | Chapter 25 JShell discussions that are designed to be covered after specific earlier chapters. (Part 1 of 2.)

JShell discussions	Can be covered after
<p>Section 25.7 presents additional JShell auto-completion capabilities for experimentation and discovery, including viewing method parameters, documentation and method overloads.</p> <p>Section 25.8 shows how to declare and use methods in JShell, including forward referencing a method that does not yet exist in the JShell session.</p>	Chapter 5, Methods
Section 25.9 shows how exceptions are handled in JShell.	Chapter 6, Arrays and ArrayLists
Section 25.10 shows how to add existing packages to the classpath and import them for use in JShell.	Chapter 21, Custom Generic Data Structures
<p>The remaining JShell sections are reference material that can be covered after Section 25.10. Topics include using an external editor, a summary of JShell commands, getting help in JShell, additional features of <code>/edit</code> command, <code>/reload</code> command, <code>/drop</code> command, feedback modes, other JShell features configurable with <code>/set</code>, keyboard shortcuts for snippet editing, how JShell reinterprets Java for interactive use and IDE JShell support.</p>	

Fig. 3 | Chapter 25 JShell discussions that are designed to be covered after specific earlier chapters. (Part 2 of 2.)

9 *New Chapter—The Java Module System and Other Java 9 Topics*

Because Java 9 was still under development when this book was published, we included an online chapter on the book’s Companion Website that discusses Java 9’s module system and various other Java 9 topics. This **online content will be available before Fall 2017 courses**.

Object-Oriented Programming (Part 3)

Object-oriented programming. We use a **late objects approach**, covering programming fundamentals such as data types, variables, operators, control statements, methods and arrays in the early chapters. Then students develop their first customized classes and objects in Chapter 7. [For courses that require an **early-objects approach**, you may want to consider our sister book *Java How to Program, Early Objects, 11/e.*]

Real-world case studies. The object-oriented programming presentation in the classes chapters features Account, Time, Employee, GradeBook and Card shuffling-and-dealing case studies.

Inheritance, Interfaces, Polymorphism and Composition. We use additional real-world case studies—including class Time, an Employee class hierarchy, and a Payable interface implemented in disparate Employee and Invoice classes—to illustrate these OO concepts and explain situations in which each is preferred in building industrial-strength applications. We also explain the use of current idioms, such as “**programming to an interface not an implementation**” and “**preferring composition to inheritance.**”

Exception handling. We integrate basic exception handling beginning in Chapter 6 then present a deeper treatment in Chapter 11. Exception handling is important for building **mission-critical** and **business-critical** applications. 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” and how your code should handle those exceptions.

Classes Arrays and ArrayList. Chapter 6 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. 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 actually execute on your computer simulator. Students in first and second programming courses, respectively, enjoy these challenges.

Flexible JavaFX GUI, Graphics, Animation and Video Coverage (Part 4) and Optional Swing Coverage (Part 11)

Students enjoy building applications with **GUI, graphics, animations and videos**. Instructors teaching introductory courses can choose the amount of GUI, graphics, animation and video they’d like to cover—from none at all to the four options discussed below. Those who want to use the newer **JavaFX GUI, graphics, animation and video** capabilities (today’s most popular options for college courses and professionals) in their courses can choose from:

- a **deep treatment** of **JavaFX GUI, graphics (2D and 3D), animation and video** in Chapters 12, 13 and 22, or
- a **lighter treatment** of **JavaFX GUI and 2D graphics** on the Companion Website that can be taught anytime after Chapter 7.

Those who want to continue using the older **Swing GUI and Java 2D graphics** in their courses can choose from the following options on the Companion Website:

- a **deep treatment** of **Swing GUI and Java 2D graphics** in online Chapters 26, 27 and 35, or
- a **lighter treatment** of **Swing GUI and Java 2D graphics** that can be taught anytime after Chapter 7.

Let’s consider these four options in more detail.

Deep Treatment of JavaFX GUI, Graphics, Animation and Video in Chapters 12, 13, 22 For this 11th edition, we’ve significantly updated our **JavaFX** presentation and moved all three chapters into the print book, replacing our **Swing GUI and graphics coverage** (which is now on the book’s **Companion Website** for instructors who want to continue with **Swing**).

In Chapters 12–13, we use **JavaFX** and **Scene Builder**—a drag-and-drop tool for creating JavaFX GUIs quickly and conveniently—to build several apps that demonstrate various JavaFX GUI layouts, controls and event-handling capabilities. In **Swing**, drag-and-drop tools and their generated code are *IDE dependent*. **Scene Builder** is a standalone tool that you can use separately or with any of the Java IDEs to do **portable drag-and-drop GUI design**.

In Chapter 22, we present **JavaFX 2D and 3D graphics, animation and video** capabilities. We also provide **36 programming exercises and projects** that students will find

challenging and entertaining, including many **game-programming exercises**. **Chapter 22 can be covered immediately after Section 13.3**. We also use JavaFX in several GUI-based examples in Chapter 23, Concurrency and Chapter 24, Accessing Databases with JDBC.

Lighter Treatment of JavaFX GUI and 2D Graphics

For instructors who like to introduce a lighter treatment of JavaFX GUI and graphics earlier than Chapter 12, we've placed on the book's Companion Website a lighter case study (Fig. 4)¹ that can be taught anytime after Chapter 7. The goal is to create a simple polymorphic drawing application in which the user can select a shape to draw and the shape's characteristics (such as its color, stroke thickness and whether it's hollow or filled) then drag the mouse to position and size the shape. The case study builds gradually toward that goal, with the reader implementing a polymorphic drawing app, then adding a more robust user interface. For courses that include these case study sections, instructors can opt to cover none, some or all of the deeper treatment in Chapters 12, 13 and 22.

Part	What you'll do
1. A Simple GUI	Display text and an image.
2. Event Handling and Drawing Lines	In response to a Button click, draw lines using JavaFX graphics capabilities.
3. Drawing Rectangles and Ovals	Draw rectangles and ovals.
4. Colors and Filled Shapes	Draw filled shapes in multiple colors.
5. Drawing Arcs	Draw a rainbow of colored arcs.
6. Using Objects with Graphics	Store shapes as objects then have those objects to draw themselves on the screen.
7. Drawing with Polymorphism	Identify the similarities between shape classes and create and use a shape class hierarchy.
8. Interactive Polymorphic Drawing Application	In capstone Exercise 13.9 you'll enable users to select each shape to draw, configure its properties (such as color and fill), and drag the mouse to position and size the shape.

Fig. 4 | Optional JavaFX GUI and Graphics Case Study.

Deep Treatment Swing GUI and 2D Graphics

Swing is still widely used, but Oracle will provide only minor updates going forward. For instructors and readers who wish to continue using Swing, we've moved to the book's **Companion Website** the 10th edition's

- Chapter 26, Swing GUI Components: Part 1
- Chapter 27, Graphics and Java 2D
- Chapter 35, Swing GUI Components: Part 2.

See the "Companion Website" section later in this Preface.

1. The deeper graphics treatment in Chapter 22 uses JavaFX shape types that can be added directly to the GUI using **Scene Builder**.

Lighter Treatment of Swing GUI and 2D Graphics

We've also moved to the Companion Website the older Swing-based version of the lighter JavaFX case study in Fig. 4.

Integrating Swing GUI Components in JavaFX GUIs

If you move to JavaFX, you still can use your favorite Swing capabilities. For example, in Chapter 24, we demonstrate how to display database data in a Swing `JTable` component that's embedded in a JavaFX GUI via a JavaFX `SwingNode`. As you explore Java further, you'll see that you also can incorporate JavaFX capabilities into your Swing GUIs.

Data Structures and Generic Collections (Part 5)

Data structures presentation. The chapters of Part 5 form the core of a second programming course emphasizing data structures. We begin with generic collection class `ArrayList` in Chapter 6. Our later data structures discussions (Chapters 16–21) provide a deeper treatment of generic collections—showing how to use many additional built-in collections of the Java API.

We discuss **recursion**, which is important for many reasons including implementing tree-like, data-structure classes. For computer-science majors and students in related disciplines, we discuss **searching and sorting algorithms** for manipulating the contents of collections, and provide a friendly introduction to **Big O**—a means of describing mathematically how hard an algorithm might have to work to solve a problem. Most programmers should use the built-in searching and sorting capabilities of the collections classes.

We then show how to implement **custom generic methods and classes**, including **custom generic data structures** (this, too, is intended for computer-science majors—in industry, most programmers should use the pre-built generic collections). **Lambdas and streams** (introduced in Chapter 17) are especially useful for working with generic collections.

Flexible Lambdas and Streams Coverage (Chapter 17)

The most significant new features in Java 8 were lambdas and streams. This book has several audiences, including

- those who'd like a significant treatment of lambdas and streams
- those who want a basic introduction with a few simple examples
- those who do not want to use lambdas and streams yet.

For this reason, we've placed most of the lambdas and streams treatment in Chapter 17, which is architected as a series of *easy-to-include-or-omit* sections that are keyed to the book's earlier sections and chapters. We do integrate lambdas and streams into a few examples *after* Chapter 17, because their capabilities are so compelling.

In Chapter 17, you'll see that lambdas and streams can help you write programs faster, more concisely, more simply, with fewer bugs and that are easier to **parallelize** (to realize performance improvements on **multi-core systems**) than programs written with previous techniques. You'll see that “functional programming” with lambdas and streams complements object-oriented programming.

Many of Chapter 17's sections are written so they can be covered earlier in the book (Fig. 5)—we suggest that students begin by covering Sections 17.1–17.7 after Chapter 6 and that professionals begin by covering Sections 17.1–17.5 after Chapter 4. After reading Chapter 17, you'll be able to cleverly reimplement many examples throughout the book.

Lambdas and streams discussions	Can be covered after
Sections 17.1—17.5 introduce basic lambda and streams capabilities that you can use to replace counting loops , and discuss the mechanics of how streams are processed.	Chapter 4, Control Statements: Part 2; Logical Operators
Section 17.6 introduces method references and additional streams capabilities.	Chapter 5, Methods
Section 17.7 introduces streams capabilities that process one-dimensional arrays .	Chapter 6, Arrays and ArrayLists
Sections 17.8—17.10 demonstrate additional streams capabilities and present various functional interfaces used in streams processing .	Chapter 10, Object-Oriented Programming: Polymorphism and Interfaces— Section 10.10 introduces Java 8 interface features (default methods , static methods and the concept of functional interfaces) for the functional interfaces that support lambdas and streams.
Section 17.11 uses lambdas and streams to process collections of String objects .	Chapter 14, Strings, Characters and Regular Expressions
Section 17.12 uses lambdas and streams to process a List<Employee> .	Chapter 16, Generic Collections
Section 17.13 uses lambdas and streams to process lines of text from a file .	Chapter 15, Files, Input/Output Streams, NIO and XML Serialization
Section 17.14 introduces streams of random values	All earlier Chapter 17 sections.
Section 17.15 introduces infinite streams	All earlier Chapter 17 sections.
Section 17.16 uses lambdas to implement JavaFX event-listener interfaces .	Chapter 12, JavaFX Graphical User Interfaces: Part 1
Chapter 23, Concurrency, shows that programs using lambdas and streams are often easier to parallelize so they can take advantage of multi-core architectures to enhance performance. The chapter demonstrates parallel stream processing and shows that Arrays method parallelSort improves performance on multi-core architectures when sorting large arrays.	

Fig. 5 | Java 8 lambdas and streams discussions and examples.

Concurrency and Multi-Core Performance (Part 6)

8

We were privileged to have as a reviewer of *Java How to Program, 10/e* Brian Goetz, co-author of *Java Concurrency in Practice* (Addison-Wesley). We updated Chapter 23, Concurrency, with Java 8 technology and idiom. We added a **parallelSort** vs. **sort** example that uses the **Java 8 Date/Time API** to time each operation and demonstrate **parallelSort**'s better performance on a **multi-core** system. We included a **Java 8 parallel vs. sequential stream processing** example, again using the **Date/Time API** to show performance improvements. We added a Java 8 **CompletableFuture** example that demonstrates sequential and parallel execution of long-running calculations and we discuss **Completable-**

Future enhancements in the online Java 9 chapter. Finally, we added several new exercises, including one that demonstrates the problems with parallelizing Java 8 streams that apply non-associative operations and several that have the reader investigate and use the **Fork/Join framework** to **parallelize recursive algorithms**.

JavaFX concurrency. In this edition, we converted Chapter 23’s Swing-based GUI examples to JavaFX. We now use **JavaFX concurrency** features, including class `Task` to execute long-running tasks in separate threads and display their results in the JavaFX application thread, and the `Platform` class’s `runLater` method to schedule a `Runnable` for execution in the JavaFX application thread.

Database: JDBC and JPA (Part 7)

JDBC. Chapter 24 covers the widely used **JDBC** and uses the **Java DB database management system**. The chapter introduces **Structured Query Language (SQL)** and features a case study on developing a JavaFX database-driven address book that demonstrates **prepared statements**. In JDK 9, Oracle no longer bundles Java DB, which is simply an Oracle-branded version of Apache Derby. JDK 9 users can download and use Apache Derby instead (<https://db.apache.org/derby/>).

Java Persistence API. Chapter 29 covers the newer **Java Persistence API (JPA)**—a standard for **object relational mapping (ORM)** that uses JDBC “under the hood.” ORM tools can look at a database’s **schema** and generate a set of classes that enabled you to interact with a database without having to use JDBC and SQL directly. This speeds database-application development, reduces errors and produces more portable code.

Web Application Development and Web Services (Part 8)

Java Server Faces (JSF). Chapters 30–31 introduce the **JavaServer™ Faces (JSF)** technology for building web-based applications. Chapter 30 includes examples on building web application GUIs, validating forms and session tracking. Chapter 31 discusses data-driven JSF applications—including a **multi-tier web address book application** that allows users to add and search for contacts.

Web services. Chapter 32 now concentrates on creating and consuming **REST-based web services**. Most of today’s web services use REST, which is simpler and more flexible than older web-services technologies that often required manipulating data in only XML format. REST can use a variety of formats, such as JSON, HTML, plain text, media files and XML.

Optional Online Object-Oriented Design Case Study (Part 10)

Developing an Object-Oriented Design and Java Implementation of an ATM. Chapters 33–34 include an *optional* case study on object-oriented design using 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 **complete Java implementation**. Students often report having a “light-bulb moment”—the case study helps them “tie it all together” and understand object orientation more deeply.

Teaching Approach

Java How to Program, 11/e, contains hundreds of complete working code examples. We stress program clarity and concentrate on building well-engineered software.

Syntax Shading. For readability, we syntax shade all the Java code, similar to the way most Java integrated-development environments and code editors syntax color code. Our syntax-shading conventions are as follows:

```

comments appear in light gray like this
keywords appear bold black like this
constants and literal values appear in bold dark gray like this
all other code appears in black like this

```

Code Highlighting. We place gray rectangles around key code segments.

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. We emphasize on-screen components in the **bold Helvetica** font (e.g., the **File** menu) and emphasize Java program text in the **Lucida** font (for example, `int x = 5;`).

Objectives. The list of chapter objectives provides a high-level overview of the chapter's contents.

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

Summary Bullets. We present a section-by-section bullet-list summary of the chapter. For ease of reference, we generally include the page number of each key term's defining occurrence in the text.

Self-Review Exercises and Answers. Extensive self-review exercises *and* answers are included for self study. All of the exercises in the optional ATM case study are fully solved.

Exercises. The chapter exercises include:

- simple recall of important terminology and concepts
- What's wrong with this code?
- What does this code do?
- writing individual statements and small portions of methods and classes
- writing complete methods, classes and programs
- major projects
- in many chapters, **Making a Difference** exercises that encourage you to use computers and the Internet to research and address significant social problems.
- In this edition, we added new exercises to our **game-programming** set (**SpotOn**, **Horse Race**, **Cannon**, **15 Puzzle**, **Hangman**, **Block Breaker**, **Snake** and **Word Search**), as well as others on the **JavaMoney API**, `final` instance variables, combining composition and inheritance, working with interfaces, drawing fractals, recursively searching directories, visualizing sorting algorithms and implementing parallel recursive algorithms with the `Fork/Join` framework. Many of these require students to research additional Java features online and use them.

Index. We've included an extensive index. Defining occurrences of key terms are highlighted with a **bold** page number. The print book index mentions only those terms used in the print book. **The online chapters index on the Companion Website includes all the print book terms and the online chapter terms.**

Programming Wisdom

We include hundreds of programming tips to help you focus on important aspects of program development. These represent the best we've gleaned from a combined nine 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 bugs and removing them from your programs; many describe aspects of Java that prevent bugs from getting into programs in the first place.



Performance Tip

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.

What are JEPs, JSRs and the JCP?

Throughout the book we encourage you to research various aspects of Java online. Some acronyms you're likely to see are JEP, JSR and JCP.

JEPs (JDK Enhancement Proposals) are used by Oracle to gather proposals from the Java community for changes to the Java language, APIs and tools, and to help create the roadmaps for future Java Standard Edition (Java SE), Java Enterprise Edition (Java EE)

and Java Micro Edition (Java ME) platform versions and the JSRs (Java Specification Requests) that define them. The complete list of JEPs can be found at

<http://openjdk.java.net/jeps/0>

JSRs (Java Specification Requests) are the formal descriptions of Java platform features' technical specifications. Each new feature that gets added to Java (Standard Edition, Enterprise Edition or Micro Edition) has a JSR that goes through a review and approval process before the feature is added to Java. Sometimes JSRs are grouped together into an umbrella JSR. For example JSR 337 is the umbrella for Java 8 features, and JSR 379 is the umbrella for Java 9 features. The complete list of JSRs can be found at

<https://www.jcp.org/en/jsr/all>

The **JCP (Java Community Process)** is responsible for developing JSRs. JCP expert groups create the JSRs, which are publicly available for review and feedback. You can learn more about the JCP at:

<https://www.jcp.org>

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 audited our book against the CERT Oracle Secure Coding Standard for Java

<http://bit.ly/CERTOracleSecureJava>

and adhered to various secure coding practices as appropriate for a textbook at this level.

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 by employing programming practices that prevent system attacks from succeeding.

We'd like to thank Robert C. Seacord. A few years back, when Mr. Seacord was the Secure Coding Manager at CERT and an adjunct professor in the Carnegie Mellon University School of Computer Science, he was a technical reviewer for our book, *C How to Program, 7/e*, where he scrutinized our C programs from a security standpoint, recommending that we adhere to the *CERT C Secure Coding Standard*. This experience also influenced our coding practices in *C++ How to Program, 10/e* and *Java How to Program, 11/e*.

Companion Website: Source Code, VideoNotes, Online Chapters and Online Appendices

All the source code for the book's code examples is available at the book's **Companion Website**, which also contains extensive **VideoNotes** and the online chapters and appendices:

www.pearsonglobaleditions.com

See the book's inside front cover for information on accessing the Companion Website.

In the extensive **VideoNotes**, co-author Paul Deitel patiently explains most of the programs in the book's core chapters. Students like viewing the VideoNotes for reinforcement of core concepts and for additional insights.

Software Used in *Java How to Program, 11/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 using the free Java Standard Edition Development Kit (JDK) 8. For the optional Java 9 content, we used the OpenJDK's early access version of JDK 9. All of the Java 9 programs run on the early access versions of JDK 9. All of the programs were tested on Windows, macOS and Linux. Several online chapters use the Netbeans IDE.

8
9

Java Documentation Links

Throughout the book, we provide links to Java documentation where you can learn more about various topics that we present. For Java 8 documentation, the links begin with

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

and for Java 9 documentation, the links currently begin with

```
http://download.java.net/java/jdk9/
```

The Java 9 documentation links will change when Oracle releases Java 9—*possibly* to links beginning with

```
http://docs.oracle.com/javase/9/
```

8
9

Java How to Program, Early Objects Version, 11/e

There are several approaches to teaching first courses in Java programming. The two most popular are the **late objects approach** and the **early objects approach**. To meet these diverse needs, we've published two versions of this book:

- *Java How to Program, Late Objects Version, 11/e* (this book), and
- *Java How to Program, Early Objects Version, 11/e*

The key difference between them is the order in which we present Chapters 1–7. The books have identical content in Chapter 8 and higher. Instructors can request an examination copy of either of these books from their Pearson representative.

Instructor Supplements

The following supplements are available to qualified instructors only through Pearson Education's Instructor Resource Center (www.pearsonglobal editions.com):

- *PowerPoint*[®] *slides* containing all the code and figures in the text, plus bulleted items that summarize key points.
- *Test Item File* of multiple-choice questions and answers (approximately two per book section).
- *Solutions Manual* with solutions to most of the end-of-chapter exercises. **Before assigning an exercise for homework, instructors should check the IRC to be sure it includes the solution. Solutions are *not* provided for “project” exercises.**

Please do not write to us requesting access to the Pearson Instructor's Resource Center which contains the book's instructor supplements, including the exercise solutions. Access is limited strictly to college instructors teaching from the book. Instructors may obtain access only through their Pearson representatives. Solutions are *not* provided for “project” exercises. If you're not a registered faculty member, contact your Pearson representative.

Acknowledgments

We'd like to thank Barbara Deitel for long hours devoted to technical research on this project. We're fortunate to have worked with the dedicated team of publishing professionals at Pearson. We appreciate the guidance, wisdom and energy of Tracy Johnson, Executive Editor, Computer Science. Tracy and her team handle all of our academic textbooks. Kristy Alaura recruited the book's reviewers and managed the review process. Bob Engelhardt managed the book's publication. We selected the cover art and Chuti Prasertsith designed the cover.

Reviewers

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

We appreciate the guidance of JavaFX experts Jim Weaver and Johan Vos (co-authors of *Pro JavaFX 8*), Jonathan Giles and Simon Ritter on the three JavaFX chapters.

Eleventh Edition reviewers: Marty Allen (University of Wisconsin-La Crosse), Robert Field (JShell chapter only; JShell Architect, Oracle), Trisha Gee (JetBrains, Java Champion), Jonathan Giles (Consulting Member of Technical Staff, Oracle), Brian Goetz (JShell chapter only; Oracle’s Java Language Architect), Edwin Harris (M.S. Instructor at The University of North Florida’s School of Computing), Maurice Naftalin (Java Champion), José Antonio González Seco (Consultant), Bruno Souza (President of SouJava—the Brazilian Java Society, Java Specialist at ToolsCloud, Java Champion and SouJava representative at the Java Community Process), Dr. Venkat Subramaniam (President, Agile Developer, Inc. and Instructional Professor, University of Houston, Java Champion), Johan Vos (CTO, Cloud Products at Gluon, Java Champion).

Tenth Edition reviewers: Lance Andersen (Oracle Corporation), Dr. Danny Coward (Oracle Corporation), Brian Goetz (Oracle Corporation), Evan Golub (University of Maryland), Dr. Huiwei 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*).

Earlier 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 Andalusia), 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 Robert Field

Robert Field, Oracle’s JShell Architect reviewed the new JShell chapter, responding to our numerous emails in which we asked JShell questions, reported bugs we encountered as JShell evolved and suggested improvements. It was a privilege having our content scrutinized by the person responsible for JShell.

A Special Thank You to Brian Goetz

Brian Goetz, Oracle’s Java Language Architect and Specification Lead for Java 8’s Project Lambda, and co-author of *Java Concurrency in Practice*, did a full-book review of the 10th edition. He provided us with an extraordinary collection of insights and constructive comments. For the 11th edition, he did a detailed review of our new JShell chapter and answered our Java questions throughout the project.

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

deitel@deitel.com

We’ll respond promptly. We hope you enjoy working with *Java How to Program, 11/e*, as much as we enjoyed researching and writing it!

Paul and Harvey Deitel

About the Authors

Paul J. Deitel, CEO and Chief Technical Officer of Deitel & Associates, Inc., is a graduate of MIT and has over 35 years of experience in computing. 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 (now Oracle), 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 M. Deitel, Chairman and Chief Strategy Officer of Deitel & Associates, Inc., has over 55 years of experience in computing. Dr. Deitel earned B.S. and M.S. degrees in Electrical Engineering from MIT and a Ph.D. in Mathematics from Boston University—he studied computing in each of these programs before they spun off Computer Science programs. 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 more than 100 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 academic, corporate, 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, Swift and iOS app development, C++, C, Visual C#®, Visual Basic®, object technology, Internet and web programming and a growing list of additional programming and software development courses.

Through its 42-year publishing partnership with Pearson/Prentice Hall, Deitel & Associates, Inc., publishes leading-edge programming textbooks and professional books in print and e-book formats, **LiveLessons** video courses and **REVEL™** online interactive multimedia courses with integrated **MyProgrammingLab**. 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, write to

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

<http://www.informit.com/store/sales.aspx>

Acknowledgments for the Global Edition

Pearson would like to thank and acknowledge the following people for their contribution to the Global Edition.

Contributor: Muthuraj M.

Reviewers: Annette Bieniusa (University of Kaiserslautern), Bogdan Oancea (University of Bucharest), and Shaligram Prajapat (Devi Ahilya University)



Before You Begin

This section contains information you should review before using this book. In addition, we provide getting-started videos that demonstrate the instructions in this Before You Begin section.

Font and Naming Conventions

We use fonts to distinguish between on-screen components (such as menu names and menu items) and Java code or commands. Our convention is to emphasize on-screen components in a sans-serif bold **Helvetica** font (for example, **File** menu) and to emphasize Java code and commands in a sans-serif *Lucida* font (for example, `System.out.println()`).

Java SE Development Kit (JDK)

The software you'll need for this book is available free for download from the web. Most of the examples were tested with the Java SE Development Kit 8 (also known as JDK 8). The most recent JDK version is available from:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

The current version of the JDK at the time of this writing is JDK 8 update 121.

Java SE 9

The Java SE 9-specific features that we discuss in optional sections and chapters require JDK 9. At the time of this writing, JDK 9 was available as an early access version. If you're using this book before the final JDK 9 is released, see the section "Installing and Configuring JDK 9 Early Access Version" later in this Before You Begin. We also discuss in that section how you can manage multiple JDK versions on Windows, macOS and Linux.

JDK Installation Instructions

After downloading the JDK installer, be sure to carefully follow the installation instructions for your platform at:

https://docs.oracle.com/javase/8/docs/technotes/guides/install/install_overview.html

You'll need to update the JDK version number in any version-specific instructions. For example, the instructions refer to `jdk1.8.0`, but the current version at the time of this writing is `jdk1.8.0_121`. If you're a Linux user, your distribution's software package manager

might provide an easier way to install the JDK. For example, you can learn how to install the JDK on Ubuntu here:

<http://askubuntu.com/questions/464755/how-to-install-openjdk-8-on-14-04-lts>

Setting the PATH Environment Variable

The PATH environment variable on your computer designates which directories the computer searches when looking for applications, such as the applications that enable you to compile and run your Java applications (called `javac` and `java`, respectively). *Carefully follow the installation instructions for Java on your platform to ensure that you set the PATH environment variable correctly.* The steps for setting environment variables differ by operating system. Instructions for various platforms are listed at:

<http://www.java.com/en/download/help/path.xml>

If you do not set the PATH variable correctly on Windows and some Linux installations, when you use the JDK's tools, you'll receive a message like:

```
'java' is not recognized as an internal or external command,
operable program or batch file.
```

In this case, go back to the installation instructions for setting the PATH and recheck your steps. If you've downloaded a newer version of the JDK, you may need to change the name of the JDK's installation directory in the PATH variable.

JDK Installation Directory and the `bin` Subdirectory

The JDK's installation directory varies by platform. The directories listed below are for Oracle's JDK 8 update 121:

- JDK on Windows:
C:\Program Files\Java\jdk1.8.0_121
- macOS (formerly called OS X):
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home
- Ubuntu Linux:
/usr/lib/jvm/java-8-oracle

Depending on your platform, the JDK installation folder's name might differ if you're using a different JDK 8 update. For Linux, the install location depends on the installer you use and possibly the Linux version as well. We used Ubuntu Linux. The PATH environment variable must point to the JDK installation directory's `bin` subdirectory.

When setting the PATH, be sure to use the proper JDK-installation-directory name for the specific version of the JDK you installed—as newer JDK releases become available, the JDK-installation-directory name changes with a new *update version number*. For example, at the time of this writing, the most recent JDK 8 release was update 121. For this version, the JDK-installation-directory name typically ends with `_121`.

CLASSPATH Environment Variable

If you attempt to run a Java program and receive a message like

```
Exception in thread "main" java.lang.NoClassDefFoundError: YourClass
```