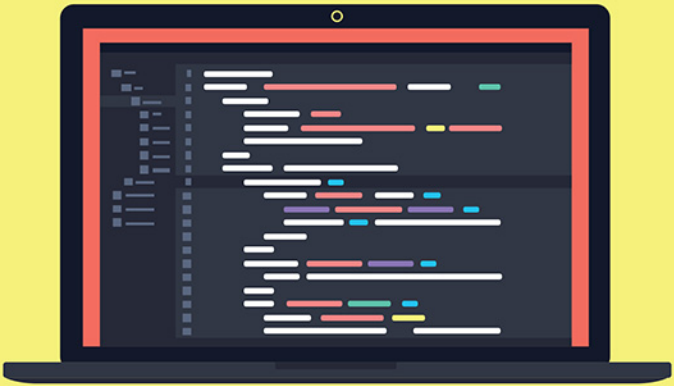
GLOBAL EDITION

Java™ How to Program

Early Objects

ELEVENTH EDITION

Paul Deitel • Harvey Deitel



Pearson

Java™

HOW TO PROGRAM

EARLY OBJECTS

ELEVENTH EDITION

GLOBAL EDITION

This page intentionally left blank.

Java™

HOW TO PROGRAM

EARLY OBJECTS

ELEVENTH EDITION
GLOBAL EDITION

Paul Deitel
Deitel & Associates, Inc.

Harvey Deitel
Deitel & Associates, Inc.

Pearson

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***
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: ***Trudy Kimber***
Media Production Manager, Global Edition: ***Vikram Kumar***
Inventory Manager: ***Ann Lam***
Product Marketing Manager: ***Yvonne Vannatta***
Field Marketing Manager: ***Demetrius Hall***
Marketing Assistant: ***Jon Bryant***
Cover Designer: ***Lumina Datamatics***
Cover Art: ***©MchlSkhrv/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 9NA
United Kingdom

and Associated Companies throughout the world

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

Authorized adaptation from the United States edition, entitled Java How to Program,Early Objects, 11th Edition, ISBN 978-0-13-474335-6 by Paul Deitel and Harvey Deitel published by Pearson Education © 2018.

British Library Cataloguing-in-Publication Data

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

10 9 8 7 6 5 4 3 2 1

ISBN 10: 1-292-22385-5
ISBN 13: 978-1-292-22385-8

Typeset by GEX Publishing Services
Printed and bound in Malaysia

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/deitel`)—see the inside front cover of your book for details.

Foreword 25

Preface 27

Before You Begin 47

1 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 Introduction to Object Technology 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 Methopd 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

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 `printf` 96
2.5 Another Application: Adding Integers 97
2.5.1 `import` 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 Introduction to Classes, Objects, Methods and Strings 120

3.1 Introduction 121
3.2 Instance Variables, *set* Methods and *get* Methods 122
3.2.1 `Account` Class with an Instance Variable, and *set* and *get* Methods 122
3.2.2 `AccountTest` Class That Creates and Uses an Object of Class `Account` 125

3.2.3 Compiling and Executing an App with Multiple Classes 128
3.2.4 `Account` UML Class Diagram 128
3.2.5 Additional Notes on Class `AccountTest` 130
3.2.6 Software Engineering with `private` Instance Variables and `public` *set* and *get* Methods 130
3.3 `Account` Class: Initializing Objects with Constructors 131
3.3.1 Declaring an `Account` Constructor for Custom Object Initialization 132
3.3.2 Class `AccountTest`: Initializing `Account` Objects When They're Created 133
3.4 `Account` Class with a Balance; Floating-Point Numbers 134
3.4.1 `Account` Class with a `balance` Instance Variable of Type `double` 135
3.4.2 `AccountTest` Class to Use Class `Account` 137
3.5 Primitive Types vs. Reference Types 140
3.6 (Optional) GUI and Graphics Case Study: A Simple GUI 140
3.6.1 What Is a Graphical User Interface? 142
3.6.2 JavaFX Scene Builder and FXML 142
3.6.3 **Welcome** App—Displaying Text and an Image 142
3.6.4 Opening Scene Builder and Creating the File `Welcome.fxml` 142
3.6.5 Adding an Image to the Folder Containing `Welcome.fxml` 144
3.6.6 Creating a `VBox` Layout Container 144
3.6.7 Configuring the `VBox` 144
3.6.8 Adding and Configuring a `Label` 144
3.6.9 Adding and Configuring an `ImageView` 146
3.6.10 Previewing the **Welcome** GUI 147
3.7 Wrap-Up 148

4 Control Statements: Part 1; Assignment, ++ and -- Operators 156

4.1 Introduction 157
4.2 Algorithms 157
4.3 Pseudocode 158
4.4 Control Structures 158
4.4.1 Sequence Structure in Java 159
4.4.2 Selection Statements in Java 160
4.4.3 Iteration Statements in Java 160
4.4.4 Summary of Control Statements in Java 160
4.5 `if` Single-Selection Statement 161
4.6 `if...else` Double-Selection Statement 162
4.6.1 Nested `if...else` Statements 163
4.6.2 Dangling-`else` Problem 164
4.6.3 Blocks 164
4.6.4 Conditional Operator (`?:`) 165
4.7 `Student` Class: Nested `if...else` Statements 165
4.8 `while` Iteration Statement 168
4.9 Formulating Algorithms: Counter-Controlled Iteration 170

4.10 Formulating Algorithms: Sentinel-Controlled Iteration 174
4.11 Formulating Algorithms: Nested Control Statements 181
4.12 Compound Assignment Operators 185
4.13 Increment and Decrement Operators 186
4.14 Primitive Types 189
4.15 (Optional) GUI and Graphics Case Study: Event Handling; Drawing Lines 190
4.15.1 Test-Driving the Completed **Draw Lines** App 190
4.15.2 Building the App's GUI 191
4.15.3 Preparing to Interact with the GUI Programmatically 195
4.15.4 Class `DrawLinesController` 197
4.15.5 Class `DrawLines`—The Main Application Class 199
4.16 Wrap-Up 201

5 Control Statements: Part 2; Logical Operators 216

5.1 Introduction 217
5.2 Essentials of Counter-Controlled Iteration 217
5.3 `for` Iteration Statement 218
5.4 Examples Using the `for` Statement 222
5.4.1 Application: Summing the Even Integers from 2 to 20 223
5.4.2 Application: Compound-Interest Calculations 224
5.5 `do...while` Iteration Statement 227
5.6 `switch` Multiple-Selection Statement 228
5.7 Class `AutoPolicy` Case Study: `String`s in `switch` Statements 234
5.8 `break` and `continue` Statements 237
5.8.1 `break` Statement 237
5.8.2 `continue` Statement 238
5.9 Logical Operators 239
5.9.1 Conditional AND (`&&`) Operator 239
5.9.2 Conditional OR (`||`) Operator 240
5.9.3 Short-Circuit Evaluation of Complex Conditions 241
5.9.4 Boolean Logical AND (`&`) and Boolean Logical Inclusive OR (`|`) Operators 241
5.9.5 Boolean Logical Exclusive OR (`^`) 242
5.9.6 Logical Negation (`!`) Operator 242
5.9.7 Logical Operators Example 243
5.10 Structured-Programming Summary 245
5.11 (Optional) GUI and Graphics Case Study: Drawing Rectangles and Ovals 250
5.12 Wrap-Up 253

6 Methods: A Deeper Look 264

6.1 Introduction 265
6.2 Program Units in Java 265
6.3 `static` Methods, `static` Fields and Class `Math` 267

6.4 Methods with Multiple Parameters 269
6.5 Notes on Declaring and Using Methods 273
6.6 Method-Call Stack and Activation Records 274
6.6.1 Method-Call Stack 274
6.6.2 Stack Frames 274
6.6.3 Local Variables and Stack Frames 274
6.6.4 Stack Overflow 275
6.7 Argument Promotion and Casting 275
6.8 Java API Packages 276
6.9 Case Study: Secure Random-Number Generation 278
6.10 Case Study: A Game of Chance; Introducing `enum` Types 283
6.11 Scope of Declarations 288
6.12 Method Overloading 290
6.12.1 Declaring Overloaded Methods 290
6.12.2 Distinguishing Between Overloaded Methods 291
6.12.3 Return Types of Overloaded Methods 292
6.13 (Optional) GUI and Graphics Case Study: Colors and Filled Shapes 292
6.14 Wrap-Up 295

7 Arrays and `ArrayLists` 309

7.1 Introduction 310
7.2 Arrays 311
7.3 Declaring and Creating Arrays 312
7.4 Examples Using Arrays 314
7.4.1 Creating and Initializing an Array 314
7.4.2 Using an Array Initializer 315
7.4.3 Calculating the Values to Store in an Array 316
7.4.4 Summing the Elements of an Array 317
7.4.5 Using Bar Charts to Display Array Data Graphically 317
7.4.6 Using the Elements of an Array as Counters 319
7.4.7 Using Arrays to Analyze Survey Results 320
7.5 Exception Handling: Processing the Incorrect Response 322
7.5.1 The `try` Statement 322
7.5.2 Executing the `catch` Block 322
7.5.3 `toString` Method of the Exception Parameter 323
7.6 Case Study: Card Shuffling and Dealing Simulation 323
7.7 Enhanced `for` Statement 328
7.8 Passing Arrays to Methods 329
7.9 Pass-By-Value vs. Pass-By-Reference 331
7.10 Case Study: Class `GradeBook` Using an Array to Store Grades 332
7.11 Multidimensional Arrays 337
7.11.1 Arrays of One-Dimensional Arrays 338
7.11.2 Two-Dimensional Arrays with Rows of Different Lengths 338
7.11.3 Creating Two-Dimensional Arrays with Array-Creation Expressions 339

7.11.4 Two-Dimensional Array Example: Displaying Element Values 339
7.11.5 Common Multidimensional-Array Manipulations Performed with `for` Statements 340
7.12 Case Study: Class `GradeBook` Using a Two-Dimensional Array 341
7.13 Variable-Length Argument Lists 347
7.14 Using Command-Line Arguments 348
7.15 Class `Arrays` 350
7.16 Introduction to Collections and Class `ArrayList` 353
7.17 (Optional) GUI and Graphics Case Study: Drawing Arcs 357
7.18 Wrap-Up 360

8 Classes and Objects: A Deeper Look 381

8.1 Introduction 382
8.2 `Time` Class Case Study 382
8.3 Controlling Access to Members 387
8.4 Referring to the Current Object's Members with the `this` Reference 388
8.5 `Time` Class Case Study: Overloaded Constructors 390
8.6 Default and No-Argument Constructors 395
8.7 Notes on *Set* and *Get* Methods 396
8.8 Composition 397
8.9 `enum` Types 400
8.10 Garbage Collection 403
8.11 `static` Class Members 403
8.12 `static` Import 407
8.13 `final` Instance Variables 408
8.14 Package Access 409
8.15 Using `BigDecimal` for Precise Monetary Calculations 410
8.16 (Optional) GUI and Graphics Case Study: Using Objects with Graphics 413
8.17 Wrap-Up 417

9 Object-Oriented Programming: Inheritance 425

9.1 Introduction 426
9.2 Superclasses and Subclasses 427
9.3 `protected` Members 429
9.4 Relationship Between Superclasses and Subclasses 430
9.4.1 Creating and Using a `CommissionEmployee` Class 430
9.4.2 Creating and Using a `BasePlusCommissionEmployee` Class 435
9.4.3 Creating a `CommissionEmployee`–`BasePlusCommissionEmployee` Inheritance Hierarchy 440
9.4.4 `CommissionEmployee`–`BasePlusCommissionEmployee` Inheritance Hierarchy Using `protected` Instance Variables 443
9.4.5 `CommissionEmployee`–`BasePlusCommissionEmployee` Inheritance Hierarchy Using `private` Instance Variables 446
9.5 Constructors in Subclasses 450

9.6 Class `Object` 451
9.7 Designing with Composition vs. Inheritance 452
9.8 Wrap-Up 454

10 Object-Oriented Programming: Polymorphism and Interfaces 459

10.1 Introduction 460
10.2 Polymorphism Examples 462
10.3 Demonstrating Polymorphic Behavior 463
10.4 Abstract Classes and Methods 465
10.5 Case Study: Payroll System Using Polymorphism 468
10.5.1 Abstract Superclass `Employee` 469
10.5.2 Concrete Subclass `SalariedEmployee` 471
10.5.3 Concrete Subclass `HourlyEmployee` 473
10.5.4 Concrete Subclass `CommissionEmployee` 474
10.5.5 Indirect Concrete Subclass `BasePlusCommissionEmployee` 476
10.5.6 Polymorphic Processing, Operator `instanceof` and Downcasting 477
10.6 Allowed Assignments Between Superclass and Subclass Variables 482
10.7 `final` Methods and Classes 482
10.8 A Deeper Explanation of Issues with Calling Methods from Constructors 483
10.9 Creating and Using Interfaces 484
10.9.1 Developing a `Payable` Hierarchy 486
10.9.2 Interface `Payable` 487
10.9.3 Class `Invoice` 487
10.9.4 Modifying Class `Employee` to Implement Interface `Payable` 489
10.9.5 Using Interface `Payable` to Process `Invoices` and `Employees` Polymorphically 491
10.9.6 Some Common Interfaces of the Java API 492
10.10 Java SE 8 Interface Enhancements 493
10.10.1 `default` Interface Methods 493
10.10.2 `static` Interface Methods 494
10.10.3 Functional Interfaces 494
10.11 Java SE 9 `private` Interface Methods 495
10.12 `private` Constructors 495
10.13 Program to an Interface, Not an Implementation 496
10.13.1 Implementation Inheritance Is Best for Small Numbers of Tightly Coupled Classes 496
10.13.2 Interface Inheritance Is Best for Flexibility 496
10.13.3 Rethinking the Employee Hierarchy 497
10.14 (Optional) GUI and Graphics Case Study: Drawing with Polymorphism 498
10.15 Wrap-Up 500

11 Exception Handling: A Deeper Look 507

11.1 Introduction 508

11.2 Example: Divide by Zero without Exception Handling 509
11.3 Example: Handling `ArithmeticExceptions` and `InputMismatchExceptions` 511
11.4 When to Use Exception Handling 517
11.5 Java Exception Hierarchy 517
11.6 `finally` Block 521
11.7 Stack Unwinding and Obtaining Information from an Exception 525
11.8 Chained Exceptions 528
11.9 Declaring New Exception Types 530
11.10 Preconditions and Postconditions 531
11.11 Assertions 531
11.12 `try`-with-Resources: Automatic Resource Deallocation 533
11.13 Wrap-Up 534

12 JavaFX Graphical User Interfaces: Part 1 540

12.1 Introduction 541
12.2 JavaFX Scene Builder 542
12.3 JavaFX App Window Structure 543
12.4 **Welcome** App—Displaying Text and an Image 544
12.4.1 Opening Scene Builder and Creating the File `Welcome.fxml` 544
12.4.2 Adding an Image to the Folder Containing `Welcome.fxml` 545
12.4.3 Creating a `VBox` Layout Container 545
12.4.4 Configuring the `VBox` Layout Container 546
12.4.5 Adding and Configuring a `Label` 546
12.4.6 Adding and Configuring an `ImageView` 547
12.4.7 Previewing the `Welcome` GUI 549
12.5 **Tip Calculator** App—Introduction to Event Handling 549
12.5.1 Test-Driving the **Tip Calculator** App 550
12.5.2 Technologies Overview 551
12.5.3 Building the App's GUI 553
12.5.4 `TipCalculator` Class 560
12.5.5 `TipCalculatorController` Class 562
12.6 Features Covered in the Other JavaFX Chapters 567
12.7 Wrap-Up 567

13 JavaFX GUI: Part 2 575

13.1 Introduction 576
13.2 Laying Out Nodes in a Scene Graph 576
13.3 **Painter** App: `RadioButtons`, Mouse Events and Shapes 578
13.3.1 Technologies Overview 578
13.3.2 Creating the `Painter.fxml` File 580
13.3.3 Building the GUI 580
13.3.4 `Painter` Subclass of `Application` 583
13.3.5 `PainterController` Class 584

13.4 **Color Chooser** App: Property Bindings and Property Listeners 588
13.4.1 Technologies Overview 588
13.4.2 Building the GUI 589
13.4.3 `ColorChooser` Subclass of `Application` 591
13.4.4 `ColorChooserController` Class 592
13.5 **Cover Viewer** App: Data-Driven GUIs with JavaFX Collections 594
13.5.1 Technologies Overview 595
13.5.2 Adding Images to the App's Folder 595
13.5.3 Building the GUI 595
13.5.4 CoverViewer Subclass of `Application` 597
13.5.5 `CoverViewerController` Class 597
13.6 **Cover Viewer** App: Customizing `ListView` Cells 599
13.6.1 Technologies Overview 600
13.6.2 Copying the `CoverViewer` App 600
13.6.3 `ImageTextCell` Custom Cell Factory Class 601
13.6.4 `CoverViewerController` Class 602
13.7 Additional JavaFX Capabilities 603
13.8 JavaFX 9: Java SE 9 JavaFX Updates 605
13.9 Wrap-Up 607

14 Strings, Characters and Regular Expressions 616

14.1 Introduction 617
14.2 Fundamentals of Characters and Strings 617
14.3 Class `String` 618
14.3.1 `String` Constructors 618
14.3.2 `String` Methods `length`, `charAt` and `getChars` 619
14.3.3 Comparing Strings 621
14.3.4 Locating Characters and Substrings in Strings 625
14.3.5 Extracting Substrings from Strings 627
14.3.6 Concatenating Strings 628
14.3.7 Miscellaneous `String` Methods 629
14.3.8 `String` Method `valueOf` 630
14.4 Class `StringBuilder` 631
14.4.1 `StringBuilder` Constructors 632
14.4.2 `StringBuilder` Methods `length`, `capacity`, `setLength` and `ensureCapacity` 633
14.4.3 `StringBuilder` Methods `charAt`, `setCharAt`, `getChars` and `reverse` 634
14.4.4 `StringBuilder` `append` Methods 635
14.4.5 `StringBuilder` Insertion and Deletion Methods 637
14.5 Class `Character` 638
14.6 Tokenizing `Strings` 643
14.7 Regular Expressions, Class `Pattern` and Class `Matcher` 644
14.7.1 Replacing Substrings and Splitting Strings 649
14.7.2 Classes `Pattern` and `Matcher` 651
14.8 Wrap-Up 653

15 Files, Input/Output Streams, NIO and XML Serialization 664

15.1 Introduction 665
15.2 Files and Streams 665
15.3 Using NIO Classes and Interfaces to Get File and Directory Information 667
15.4 Sequential Text Files 671
15.4.1 Creating a Sequential Text File 671
15.4.2 Reading Data from a Sequential Text File 674
15.4.3 Case Study: A Credit-Inquiry Program 675
15.4.4 Updating Sequential Files 680
15.5 XML Serialization 680
15.5.1 Creating a Sequential File Using XML Serialization 680
15.5.2 Reading and Deserializing Data from a Sequential File 686
15.6 `FileChooser` and `DirectoryChooser` Dialogs 687
15.7 (Optional) Additional `java.io` Classes 693
15.7.1 Interfaces and Classes for Byte-Based Input and Output 693
15.7.2 Interfaces and Classes for Character-Based Input and Output 695
15.8 Wrap-Up 696

16 Generic Collections 704

16.1 Introduction 705
16.2 Collections Overview 705
16.3 Type-Wrapper Classes 707
16.4 Autoboxing and Auto-Unboxing 707
16.5 Interface `Collection` and Class `Collections` 707
16.6 Lists 708
16.6.1 `ArrayList` and `Iterator` 709
16.6.2 `LinkedList` 711
16.7 `Collections` Methods 716
16.7.1 Method `sort` 716
16.7.2 Method `shuffle` 720
16.7.3 Methods `reverse`, `fill`, `copy`, `max` and `min` 722
16.7.4 Method `binarySearch` 724
16.7.5 Methods `addAll`, `frequency` and `disjoint` 725
16.8 Class `PriorityQueue` and Interface `Queue` 727
16.9 Sets 728
16.10 Maps 731
16.11 Synchronized Collections 735
16.12 Unmodifiable Collections 735
16.13 Abstract Implementations 736
16.14 Java SE 9: Convenience Factory Methods for Immutable Collections 736
16.15 Wrap-Up 740

17 Lambdas and Streams 746

17.1 Introduction 747
17.2 Streams and Reduction 749
17.2.1 Summing the Integers from 1 through 10 with a `for` Loop 749
17.2.2 External Iteration with `for` Is Error Prone 750
17.2.3 Summing with a Stream and Reduction 750
17.2.4 Internal Iteration 751
17.3 Mapping and Lambdas 752
17.3.1 Lambda Expressions 753
17.3.2 Lambda Syntax 754
17.3.3 Intermediate and Terminal Operations 755
17.4 Filtering 756
17.5 How Elements Move Through Stream Pipelines 758
17.6 Method References 759
17.6.1 Creating an `IntStream` of Random Values 760
17.6.2 Performing a Task on Each Stream Element with `forEach` and a Method Reference 760
17.6.3 Mapping Integers to String Objects with `mapToObj` 761
17.6.4 Concatenating Strings with `collect` 761
17.7 `IntStream` Operations 762
17.7.1 Creating an `IntStream` and Displaying Its Values 763
17.7.2 Terminal Operations `count`, `min`, `max`, `sum` and `average` 763
17.7.3 Terminal Operation `reduce` 764
17.7.4 Sorting `IntStream` Values 766
17.8 Functional Interfaces 767
17.9 Lambdas: A Deeper Look 768
17.10 `Stream<Integer>` Manipulations 769
17.10.1 Creating a `Stream<Integer>` 770
17.10.2 Sorting a `Stream` and Collecting the Results 771
17.10.3 Filtering a `Stream` and Storing the Results for Later Use 771
17.10.4 Filtering and Sorting a `Stream` and Collecting the Results 772
17.10.5 Sorting Previously Collected Results 772
17.11 `Stream<String>` Manipulations 772
17.11.1 Mapping `Strings` to Uppercase 773
17.11.2 Filtering `Strings` Then Sorting Them in Case-Insensitive Ascending Order 774
17.11.3 Filtering `Strings` Then Sorting Them in Case-Insensitive Descending Order 774
17.12 `Stream<Employee>` Manipulations 775
17.12.1 Creating and Displaying a `List<Employee>` 776
17.12.2 Filtering `Employees` with Salaries in a Specified Range 777
17.12.3 Sorting `Employees` By Multiple Fields 780
17.12.4 Mapping `Employees` to Unique-Last-Name `Strings` 782
17.12.5 Grouping `Employees` By Department 783
17.12.6 Counting the Number of `Employees` in Each Department 784
17.12.7 Summing and Averaging `Employee` Salaries 785

17.13 Creating a `Stream<String>` from a File 786
17.14 Streams of Random Values 789
17.15 Infinite Streams 791
17.16 Lambda Event Handlers 793
17.17 Additional Notes on Java SE 8 Interfaces 793
17.18 Wrap-Up 794

18 Recursion 808

18.1 Introduction 809
18.2 Recursion Concepts 810
18.3 Example Using Recursion: Factorials 811
18.4 Reimplementing Class `FactorialCalculator` Using `BigInteger` 813
18.5 Example Using Recursion: Fibonacci Series 815
18.6 Recursion and the Method-Call Stack 818
18.7 Recursion vs. Iteration 819
18.8 Towers of Hanoi 821
18.9 Fractals 823
18.9.1 Koch Curve Fractal 824
18.9.2 (Optional) Case Study: Lo Feather Fractal 825
18.9.3 (Optional) **Fractal** App GUI 827
18.9.4 (Optional) `FractalController` Class 829
18.10 Recursive Backtracking 834
18.11 Wrap-Up 834

19 Searching, Sorting and Big O 843

19.1 Introduction 844
19.2 Linear Search 845
19.3 Big O Notation 848
19.3.1 $O(1)$ Algorithms 848
19.3.2 $O(n)$ Algorithms 848
19.3.3 $O(n^2)$ Algorithms 848
19.3.4 Big O of the Linear Search 849
19.4 Binary Search 849
19.4.1 Binary Search Implementation 850
19.4.2 Efficiency of the Binary Search 853
19.5 Sorting Algorithms 854
19.6 Selection Sort 854
19.6.1 Selection Sort Implementation 855
19.6.2 Efficiency of the Selection Sort 857
19.7 Insertion Sort 857
19.7.1 Insertion Sort Implementation 858
19.7.2 Efficiency of the Insertion Sort 860
19.8 Merge Sort 861
19.8.1 Merge Sort Implementation 861
19.8.2 Efficiency of the Merge Sort 866

19.9 Big O Summary for This Chapter's Searching and Sorting Algorithms 866
19.10 Massive Parallelism and Parallel Algorithms 867
19.11 Wrap-Up 867

20 Generic Classes and Methods: A Deeper Look 873
20.1 Introduction 874
20.2 Motivation for Generic Methods 874
20.3 Generic Methods: Implementation and Compile-Time Translation 876
20.4 Additional Compile-Time Translation Issues: Methods That Use a Type Parameter as the Return Type 879
20.5 Overloading Generic Methods 882
20.6 Generic Classes 883
20.7 Wildcards in Methods That Accept Type Parameters 890
20.8 Wrap-Up 894

21 Custom Generic Data Structures 898
21.1 Introduction 899
21.2 Self-Referential Classes 900
21.3 Dynamic Memory Allocation 900
21.4 Linked Lists 901
21.4.1 Singly Linked Lists 901
21.4.2 Implementing a Generic `List` Class 902
21.4.3 Generic Classes `ListNode` and `List` 905
21.4.4 Class `ListTest` 905
21.4.5 `List` Method `insertAtFront` 907
21.4.6 `List` Method `insertAtBack` 908
21.4.7 `List` Method `removeFromFront` 908
21.4.8 `List` Method `removeFromBack` 909
21.4.9 `List` Method `print` 910
21.4.10 Creating Your Own Packages 910
21.5 Stacks 915
21.6 Queues 918
21.7 Trees 920
21.8 Wrap-Up 927

22 JavaFX Graphics and Multimedia 952
22.1 Introduction 953
22.2 Controlling Fonts with Cascading Style Sheets (CSS) 954
22.2.1 CSS That Styles the GUI 954
22.2.2 FXML That Defines the GUI—Introduction to XML Markup 957
22.2.3 Referencing the CSS File from FXML 960
22.2.4 Specifying the `VBox`'s Style Class 960
22.2.5 Programmatically Loading CSS 960

22.3 Displaying Two-Dimensional Shapes 961
22.3.1 Defining Two-Dimensional Shapes with FXML 961
22.3.2 CSS That Styles the Two-Dimensional Shapes 964
22.4 `Polylines`, `Polygons` and `Paths` 966
22.4.1 GUI and CSS 967
22.4.2 `PolyShapesController` Class 968
22.5 Transforms 971
22.6 Playing Video with `Media`, `MediaPlayer` and `MediaViewer` 973
22.6.1 `VideoPlayer` GUI 974
22.6.2 `VideoPlayerController` Class 976
22.7 `Transition` Animations 980
22.7.1 `TransitionAnimations.fxml` 980
22.7.2 `TransitionAnimationsController` Class 982
22.8 `Timeline` Animations 986
22.9 Frame-by-Frame Animation with `AnimationTimer` 989
22.10 Drawing on a `Canvas` 991
22.11 Three-Dimensional Shapes 996
22.12 Wrap-Up 999

23 Concurrency 1015

23.1 Introduction 1016
23.2 Thread States and Life Cycle 1018
23.2.1 *New* and *Runnable* States 1019
23.2.2 *Waiting* State 1019
23.2.3 *Timed Waiting* State 1019
23.2.4 *Blocked* State 1019
23.2.5 *Terminated* State 1019
23.2.6 Operating-System View of the *Runnable* State 1020
23.2.7 Thread Priorities and Thread Scheduling 1020
23.2.8 Indefinite Postponement and Deadlock 1021
23.3 Creating and Executing Threads with the `Executor` Framework 1021
23.4 Thread Synchronization 1025
23.4.1 Immutable Data 1026
23.4.2 Monitors 1026
23.4.3 Unsynchronized Mutable Data Sharing 1027
23.4.4 Synchronized Mutable Data Sharing—Making Operations Atomic 1031
23.5 Producer/Consumer Relationship without Synchronization 1034
23.6 Producer/Consumer Relationship: `ArrayBlockingQueue` 1042
23.7 (Advanced) Producer/Consumer Relationship with `synchronized`, `wait`, `notify` and `notifyAll` 1045
23.8 (Advanced) Producer/Consumer Relationship: Bounded Buffers 1051
23.9 (Advanced) Producer/Consumer Relationship: The `Lock` and `Condition` Interfaces 1059
23.10 Concurrent Collections 1066
23.11 Multithreading in JavaFX 1068

23.11.1 Performing Computations in a Worker Thread: Fibonacci Numbers 1069
23.11.2 Processing Intermediate Results: Sieve of Eratosthenes 1074
23.12 `sort`/`parallelSort` Timings with the Java SE 8 Date/Time API 1080
23.13 Java SE 8: Sequential vs. Parallel Streams 1083
23.14 (Advanced) Interfaces `Callable` and `Future` 1085
23.15 (Advanced) Fork/Join Framework 1090
23.16 Wrap-Up 1090

24 Accessing Databases with JDBC 1102

24.1 Introduction 1103
24.2 Relational Databases 1104
24.3 A `books` Database 1105
24.4 SQL 1109
24.4.1 Basic `SELECT` Query 1110
24.4.2 `WHERE` Clause 1110
24.4.3 `ORDER BY` Clause 1112
24.4.4 Merging Data from Multiple Tables: `INNER JOIN` 1114
24.4.5 `INSERT` Statement 1115
24.4.6 `UPDATE` Statement 1116
24.4.7 `DELETE` Statement 1117
24.5 Setting Up a Java DB Database 1118
24.5.1 Creating the Chapter's Databases on Windows 1119
24.5.2 Creating the Chapter's Databases on macOS 1120
24.5.3 Creating the Chapter's Databases on Linux 1120
24.6 Connecting to and Querying a Database 1120
24.6.1 Automatic Driver Discovery 1122
24.6.2 Connecting to the Database 1122
24.6.3 Creating a `Statement` for Executing Queries 1123
24.6.4 Executing a Query 1123
24.6.5 Processing a Query's `ResultSet` 1124
24.7 Querying the `books` Database 1125
24.7.1 `ResultSetTableModel` Class 1125
24.7.2 `DisplayQueryResults` App's GUI 1132
24.7.3 `DisplayQueryResultsController` Class 1132
24.8 `RowSet` Interface 1137
24.9 `PreparedStatements` 1140
24.9.1 `AddressBook` App That Uses `PreparedStatements` 1141
24.9.2 Class `Person` 1141
24.9.3 Class `PersonQueries` 1143
24.9.4 `AddressBook` GUI 1146
24.9.5 Class `AddressBookController` 1147
24.10 Stored Procedures 1152
24.11 Transaction Processing 1152
24.12 Wrap-Up 1153

25 Introduction to JShell: Java 9's REPL 1161

25.1 Introduction 1162
25.2 Installing JDK 9 1164
25.3 Introduction to JShell 1164
25.3.1 Starting a JShell Session 1165
25.3.2 Executing Statements 1165
25.3.3 Declaring Variables Explicitly 1166
25.3.4 Listing and Executing Prior Snippets 1168
25.3.5 Evaluating Expressions and Declaring Variables Implicitly 1170
25.3.6 Using Implicitly Declared Variables 1170
25.3.7 Viewing a Variable's Value 1171
25.3.8 Resetting a JShell Session 1171
25.3.9 Writing Multiline Statements 1171
25.3.10 Editing Code Snippets 1172
25.3.11 Exiting JShell 1175
25.4 Command-Line Input in JShell 1175
25.5 Declaring and Using Classes 1176
25.5.1 Creating a Class in JShell 1177
25.5.2 Explicitly Declaring Reference-Type Variables 1177
25.5.3 Creating Objects 1178
25.5.4 Manipulating Objects 1178
25.5.5 Creating a Meaningful Variable Name for an Expression 1179
25.5.6 Saving and Opening Code-Snippet Files 1180
25.6 Discovery with JShell Auto-Completion 1180
25.6.1 Auto-Completing Identifiers 1181
25.6.2 Auto-Completing JShell Commands 1182
25.7 Exploring a Class's Members and Viewing Documentation 1182
25.7.1 Listing Class `Math`'s `static` Members 1183
25.7.2 Viewing a Method's Parameters 1183
25.7.3 Viewing a Method's Documentation 1184
25.7.4 Viewing a `public` Field's Documentation 1184
25.7.5 Viewing a Class's Documentation 1185
25.7.6 Viewing Method Overloads 1185
25.7.7 Exploring Members of a Specific Object 1186
25.8 Declaring Methods 1188
25.8.1 Forward Referencing an Undeclared Method—Declaring Method `displayCubes` 1188
25.8.2 Declaring a Previously Undeclared Method 1188
25.8.3 Testing `cube` and Replacing Its Declaration 1189
25.8.4 Testing Updated Method `cube` and Method `displayCubes` 1189
25.9 Exceptions 1190
25.10 Importing Classes and Adding Packages to the `CLASSPATH` 1191
25.11 Using an External Editor 1193
25.12 Summary of JShell Commands 1195
25.12.1 Getting Help in JShell 1196
25.12.2 `/edit` Command: Additional Features 1197

25.12.3 `/reload` Command 1197
25.12.4 `/drop` Command 1198
25.12.5 Feedback Modes 1198
25.12.6 Other JShell Features Configurable with `/set` 1200
25.13 Keyboard Shortcuts for Snippet Editing 1201
25.14 How JShell Reinterprets Java for Interactive Use 1201
25.15 IDE JShell Support 1202
25.16 Wrap-Up 1202

Chapters on the Web 1218

A Operator Precedence Chart 1219

B ASCII Character Set 1221

C Keywords and Reserved Words 1222

D Primitive Types 1223

E Using the Debugger 1224
E.1 Introduction 1225
E.2 Breakpoints and the `run`, `stop`, `cont` and `print` Commands 1225
E.3 The `print` and `set` Commands 1229
E.4 Controlling Execution Using the `step`, `step up` and `next` Commands 1231
E.5 The `watch` Command 1233
E.6 The `clear` Command 1235
E.7 Wrap-Up 1238

Appendices on the Web 1239

Index 1241

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, Early 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

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

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.)

- Flexible JavaFX/Swing GUI, Graphics and Multimedia Coverage
- Data Structures and Generic Collections
- 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 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 (`java.time`)
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., `java.util.stream`, `java.util.function`)	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	
In the Print Book	***On the Companion Website***
New JShell chapter	Module system
_ 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, 11/e's Modular Organization[1]

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

Java How to Program, 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 3, Introduction to Classes, Objects, Methods and Strings

---

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

Part 2: Additional Programming Fundamentals
Chapter 4, Control Statements: Part 1; Assignment, `++` and `--` Operators
Chapter 5, Control Statements: Part 2; Logical Operators
Chapter 6, Methods: A Deeper Look
Chapter 7, Arrays and `ArrayLists`

Chapter 14, Strings, Characters and Regular Expressions
Chapter 15, Files, Input/Output Streams, NIO and XML Serialization

Part 3: Object-Oriented Programming
Chapter 8, Classes and Objects: A Deeper Look
Chapter 9, Object-Oriented Programming: Inheritance
Chapter 10, Object-Oriented Programming: Polymorphism and Interfaces
Chapter 11, Exception Handling: A Deeper Look

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

---

1. The online chapters and VideoNotes will be available on the book's Companion Website before Fall 2017 classes and will be updated as Java 9 evolves. Please write to `deitel@deitel.com` if you need them sooner.

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 differs from most other Java textbooks in that it features an **early 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. It's optional because not all courses will want to cover JShell. For those that do, instructors and students will appreciate how JShell's interactivity makes Java "come alive," leveraging the learning process—see the next section on JShell.

9 Flexible Coverage of Java 9: JShell, the Module System and Other Java 9 Topics (JShell Begins in Part 1; the Rest is in Part 9)

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.

We chose a modular approach with the JShell content packaged 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. Section 25.4 discusses command-line input with **`Scanner`** in JShell.	Chapter 2, Introduction to Java Applications; Input/Output and Operators
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. Section 25.6 shows how to use JShell's **auto-completion** capabilities to discover a class's capabilities and JShell commands.	Chapter 3, Introduction to Classes, Objects, Methods and Strings

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
Section 25.7 presents additional JShell auto-completion capabilities for **experimentation and discovery**, including viewing method parameters, documentation and method overloads. 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.	Chapter 6, Methods: A Deeper Look
Section 25.9 shows how **exceptions** are handled in JShell.	Chapter 7, 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
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 `/edit` command, `/reload` command, `/drop` command, feedback modes, other JShell features configurable with `/set`, keyboard shortcuts for snippet editing, how JShell reinterprets Java for interactive use and IDE JShell support.	

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 an **early objects approach**, introducing the basic concepts and terminology of object technology in Chapter 1. Students develop their first customized classes and objects in Chapter 3. Presenting objects and classes early gets students "thinking about objects" immediately and mastering these concepts more thoroughly. [For courses that require a **late-objects approach**, you may want to consider our sister book *Java How to Program, Late Objects Version, 11/e*.]

Early objects real-world case studies. The early classes and objects presentation in Chapters 3–7 features `Account`, `Student`, `AutoPolicy`, `Time`, `Employee`, `GradeBook` and `Card` shuffling-and-dealing case studies, gradually introducing deeper OO concepts.

Inheritance, Interfaces, Polymorphism and Composition. The deeper treatment of object-oriented programming in Chapters 8–10 features additional real-world case studies, including class `Time`, an `Employee` class hierarchy, and a `Payable` interface implemented in disparate `Employee` and `Invoice` classes. We explain the use of current idioms, such as "**programming to an interface not an implementation**" and "**preferring composition to inheritance**" in building industrial-strength applications.

Exception handling. We integrate basic exception handling beginning in Chapter 7 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.

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

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

For instructors teaching introductory courses, we provide a **scalable JavaFX GUI, graphics and multimedia treatment** enabling instructors to choose the amount of JavaFX they want to cover:

- from none at all,
- to some or all of the *optional* **introductory** sections at the ends of the early chapters,
- to a **deep treatment** of JavaFX GUI, graphics and multimedia in Chapters 12, 13 and 22.

We also use JavaFX in several GUI-based examples in Chapter 23, Concurrency and Chapter 24, Accessing Databases with JDBC.

Flexible Early Treatment of JavaFX

Students enjoy building applications with GUI, graphics, animations and videos. For courses that gently introduce GUI and graphics early, we've integrated an ***optional*** **GUI and Graphics Case Study** that introduces JavaFX-based graphical user interfaces (GUIs) and **`Canvas`-based graphics**.[1] The goal of this case study 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 in Chapter 10, and a more robust user interface in Exercise 13.9 (Fig. 4). For courses that include these optional early case study sections, instructors can opt to cover none, some or all of the deeper treatment in Chapters 12, 13 and 22 discussed in the next section.

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

Section or Exercise	What you'll do
Section 3.6: A Simple GUI	Display text and an image.
Section 4.15: Event Handling and Drawing Lines	In response to a `Button` click, draw lines using JavaFX graphics capabilities.
Section 5.11: Drawing Rectangles and Ovals	Draw rectangles and ovals.
Section 6.13: Colors and Filled Shapes	Draw filled shapes in multiple colors.
Section 7.17: Drawing Arcs	Draw a rainbow of colored arcs.
Section 8.16: Using Objects with Graphics	Store shapes as objects then have those objects to draw themselves on the screen.
Section 10.14: Drawing with Polymorphism	Identify the similarities between shape classes and create and use a shape class hierarchy.
Exercise 13.9: Interactive Polymorphic Drawing Application	A capstone exercise in which 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 | GUI and Graphics Case Study sections and exercise.

Deeper Treatment of JavaFX GUI, Graphics and Multimedia in Chapters 12, 13 and 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 online for instructors who want to continue with Swing). In the case study and 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 demonstrating various JavaFX 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 many 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**. Despite the fact that the JavaFX chapters are spread out in the book, **Chapter 22 can be covered immediately after Chapter 13**.

Swing GUI and Java 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

- optional Swing GUI and Graphics Case Study from Chapters 3–8, 10 and 13
- 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.

Integrating Swing GUI Components in JavaFX GUIs
Even 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 8 `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. Chapter 7 and the chapters of Part 5 form the core of a data structures course. We begin with generic class `ArrayList` in Chapter 7. Our later data structures discussions (Chapters 16–21) provide a deeper treatment of **generic collections**—showing how to use the 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 popular **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 build-in searching and sorting capabilities of the collections classes.

We then show how to implement **generic methods and classes**, and **custom generic data structures** (this, too, is intended for computer-science majors—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 17.2–17.7 after Chapter 7 and that professionals begin by covering Sections 17.2–17.5 after Chapter 5. 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 17.2–17.5 introduce basic lambda and streams capabilities that you can use to **replace counting loops**, and discuss the mechanics of how streams are processed.	Chapter 5, Control Statements: Part 2; Logical Operators
Section 17.6 introduces **method references** and additional streams capabilities.	Chapter 6, Methods: A Deeper Look
Section 17.7 introduces streams capabilities that process **one-dimensional arrays**.	Chapter 7, 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 shows how to use lambdas and streams to **process collections of `String` objects**.	Chapter 14, Strings, Characters and Regular Expressions
Section 17.12 shows how to use lambdas and streams to **process a `List<Employee>`**.	Chapter 16, Generic Collections
Section 17.13 shows how to use 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 shows how to use 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)

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 se-

quential and parallel execution of long-running calculations and we discuss **`CompletableFuture` 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**. 9

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/`). 9

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

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

Code Highlighting. We place transparent yellow 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 maroon** 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, com-

bining 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.

Exercises that focus on either Java 8 or Java 9 are marked as such. Check out our **Programming Projects Resource Center** for lots of additional exercise and project possibilities (`www.deitel.com/ProgrammingProjects/`).

Index. We've included an extensive index. Defining occurrences of key terms are highlighted with a **bold maroon** 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 Practices

The Good Programming Practices *call attention to techniques that will help you produce programs that are clearer, more understandable and more maintainable.*

Common Programming Errors

Pointing out these Common Programming Errors *reduces the likelihood that you'll make them.*

Error-Prevention Tips

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 Tips

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

Portability Tips

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

Software Engineering Observations

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

Look-and-Feel Observations

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:

```
http://www.pearsonglobaleditions.com/deitel
```

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 in *Java How to Program, 11/e*, 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 remaining programs run on both JDK 8 and early access versions of JDK 9, and were tested on Windows, macOS and Linux. Several online chapters also 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

8

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

and for Java 9 documentation, the links currently begin with

9

```
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/
```

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

```
http://www.deitel.com/books/jhtp11
```

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

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

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

```
http://www.pearsonglobaleditions.com/deitel
```

Instructor Supplements

The following supplements are available to qualified instructors only through Pearson Education's Instructor Resource Center (`www.pearsonglobaleditions.com/deitel`):

- ***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.

Online Practice and Assessment with MyProgrammingLab™

MyProgrammingLab™ helps students fully grasp the logic, semantics, and syntax of programming. Through practice exercises and immediate, personalized feedback, MyProgrammingLab improves the programming competence of beginning students who often struggle with the basic concepts and paradigms of popular high-level programming languages.

An optional self-study and homework tool, a MyProgrammingLab course consists of hundreds of small practice problems organized around the structure of this textbook. For students, the system automatically detects errors in the logic and syntax of their code submissions and offers targeted hints that enable students to figure out what went wrong—and why. For instructors, a comprehensive gradebook tracks correct and incorrect answers and stores the code inputted by students for review.

For a full demonstration, to see feedback from instructors and students or to get started using MyProgrammingLab in your course, instructors should visit

```
http://www.myprogramminglab.com
```

Keeping in Touch with the Authors

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

```
deitel@deitel.com
```

and we'll respond promptly. For **book updates**, visit

```
http://www.deitel.com/books/jhtp11
```

subscribe to the ***Deitel® Buzz Online*** **newsletter** at

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

and join the **Deitel social networking communities** on

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

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), 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. Any updates to the information presented here will be posted at:

```
http://www.deitel.com/books/jhtp11
```

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 verion. 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
```

then your system has a CLASSPATH environment variable that must be modified. To fix the preceding error, follow the steps in setting the PATH environment variable, to locate the CLASSPATH variable, then edit the variable's value to include the local directory—typically represented as a dot (.). On Windows add

```
.;
```

at the beginning of the CLASSPATH's value (with no spaces before or after these characters). On macOS and Linux, add

```
.:
```

Setting the JAVA_HOME Environment Variable

The Java DB database software that you'll use in Chapter 24 and several online chapters requires you to set the JAVA_HOME environment variable to your JDK's installation directory. The same steps you used to set the PATH may also be used to set other environment variables, such as JAVA_HOME.

Java Integrated Development Environments (IDEs)

There are many Java integrated development environments that you can use for Java programming. Because the steps for using them differ, we used only the JDK command-line tools for most of the book's examples. We provide getting-started videos that show how to download, install and use three popular IDEs—NetBeans, Eclipse and IntelliJ IDEA. We use NetBeans in several of the book's online chapters.

NetBeans Downloads

You can download the JDK/NetBeans bundle from:

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

The NetBeans version that's bundled with the JDK is for Java SE development. The online JavaServer Faces (JSF) chapters and web services chapter use the Java Enterprise Edition (Java EE) version of NetBeans, which you can download from:

```
https://netbeans.org/downloads/
```

This version supports both Java SE and Java EE development.

Eclipse Downloads

You can download the Eclipse IDE from:

```
https://eclipse.org/downloads/eclipse-packages/
```

For Java SE development choose the Eclipse IDE for Java Developers. For Java Enterprise Edition (Java EE) development (such as JSF and web services), choose the Eclipse IDE for Java EE Developers—this version supports both Java SE and Java EE development.

IntelliJ IDEA Community Edition Downloads

You can download the free IntelliJ IDEA Community from:

```
https://www.jetbrains.com/idea/download/index.html
```

The free version supports only Java SE development.