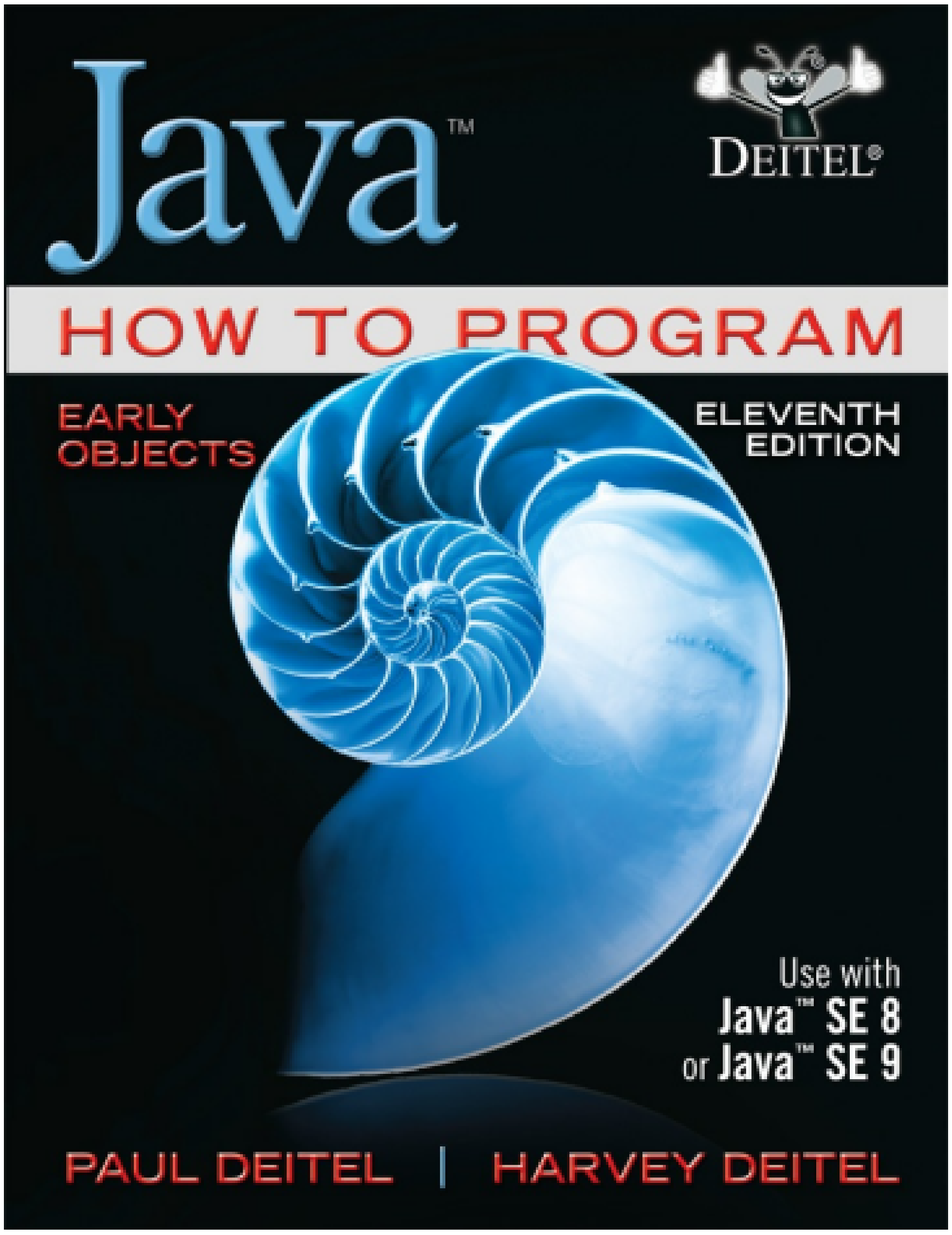
Java™
DEITEL®
HOW TO PROGRAM
EARLY OBJECTS
ELEVENTH EDITION
Use with Java™ SE 8 or Java™ SE 9
PAUL DEITEL | HARVEY DEITEL

Java™ How to Program Early Objects

ELEVENTH EDITION

Deitel Series Page

®

How To Program Series

Android™ How to Program, 3/E

C++ How to Program, 10/E

C How to Program, 8/E

Java™ How to Program, Early Objects Version,

11/E Java™ How to Program, Late Objects

Version, 11/E Internet & World Wide Web How

to Program, 5/E

Visual Basic® 2012 How to Program, 6/E

Visual C#® How to Program, 6/E

Deitel® Developer Series

Android™ 6 for Programmers: An App-Driven Approach, 3/E

C for Programmers with an Introduction to C11
C++11 for Programmers

C# 6 for Programmers

Java™ for Programmers, 4/E

JavaScript for Programmers

Swift™ for Programmers

Simply Series

Simply Visual Basic® 2010: An App-Driven Approach,

4/E Simply C++: An App-Driven Tutorial Approach

VitalSource Web Books

http://bit.ly/DeitelOnVitalSource

Android™ How to Program, 2/E and 3/E

C++ How to Program, 9/E and 10/E

Java™ How to Program, 10/E and 11/E

Simply C++: An App-Driven Tutorial Approach

Simply Visual Basic® 2010: An App-Driven Approach,

4/E ®

Visual Basic® 2012 How to Program, 6/E

Visual C#® How to Program, 6/E

Visual C#® 2012 How to Program, 5/E

LiveLessons Video Learning

Products

http://deitel.com/books/LiveLessons/ Android™ 6

App Development Fundamentals, 3/E C++

Fundamentals Java SE 8™ Fundamentals, 2/E

Java SE 9™ Fundamentals, 3/E

C# 6 Fundamentals

C# 2012 Fundamentals

JavaScript Fundamentals

Swift™ Fundamentals

REVEL™ Interactive Multimedia

REVEL™ for Deitel Java™

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

Facebook® —http://facebook.com/DeitelFan

Twitter® —http://twitter.com/deitel

LinkedIn® —http://linkedin.com/company/deitel-&- associates

YouTube™—http://youtube.com/DeitelTV

Google+™—http://google.com/+DeitelFan

Instagram® —http://instagram.com/DeitelFan

®

and register for the free *Deitel Buzz Online* e-mail newsletter at:

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

To communicate with the authors, send e-mail to:

deitel@deitel.com

For information on programming-languages corporate training seminars offered by Deitel & Associates, Inc. worldwide, write to deitel@deitel.com or visit:

http://www.deitel.com/training/

For continuing updates on Pearson/Deitel

publications visit:

http://www.deitel.com
http://www.pearsonhighered.com/deitel/

Visit the Deitel Resource Centers, which will help you master programming languages, software development, Android™ ®

and iOS app development, and Internet- and web-related topics:

http://www.deitel.com/ResourceCenters.html

Java™ How to Program Early Objects

ELEVENTH EDITION

Paul Deitel

Deitel & Associates, Inc.

Harvey Deitel

Deitel & Associates, Inc.

330 Hudson Street, NY, NY, 10013

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.pearsonhighered.com/ deitel/)**—see the inside front cover of your book for details.**

1. Foreword xxv
2. Preface xxvii
3. Before You Begin xlvii

1. 1 Introduction to Computers, the Internet and Java 1
    1. 1.1 Introduction 2
    2. 1.2 Hardware and Software 4
        1. 1.2.1 Moore's Law 4

2. 1.2.2 Computer Organization 5

3. 1.3 Data Hierarchy 7

4. 1.4 Machine Languages, Assembly Languages and High-Level Languages 9

5. 1.5 Introduction to Object Technology 10

1. 1.5.1 Automobile as an Object 11

2. 1.5.2 Methods and Classes 11

3. 1.5.3 Instantiation 11
4. 1.5.4 Reuse 11

5. 1.5.5 Messages and Methopd Calls 12

6. 1.5.6 Attributes and Instance Variables 12

7. 1.5.7 Encapsulation and Information Hiding

12 8. 1.5.8 Inheritance 12

9. 1.5.9 Interfaces 13

10. 1.5.10 Object-Oriented Analysis and Design (OOAD) 13

11. 1.5.11 The UML (Unified Modeling Language) 13

6. 1.6 Operating Systems 14

1. 1.6.1 Windows—A Proprietary Operating System 14

2. 1.6.2 Linux—An Open-Source Operating System 14

®

3. 1.6.3 Apple's macOS and Apple's iOS for iPhone , ® ®
iPad and iPod Touch Devices 15

4. 1.6.4 Google's Android 15

7. 1.7 Programming Languages 16

8. 1.8 Java 18

9. 1.9 A Typical Java Development

Environment 19 10. 1.10 Test-Driving a Java

Application 22

11. 1.11 Internet and World Wide Web 26

1. 1.11.1 Internet: A Network of Networks 27
2. 1.11.2 World Wide Web: Making the Internet User Friendly 27
3. 1.11.3 Web Services and Mashups 27
4. 1.11.4 Internet of Things 28

12. 1.12 Software Technologies 29
13. 1.13 Getting Your Questions Answered 31

2. 2 Introduction to Java Applications; Input/Output and Operators 35

1. 2.1 Introduction 36
2. 2.2 Your First Program in Java: Printing a Line of Text 36

1. 2.2.1 Compiling the Application 40
2. 2.2.2 Executing the Application 41

3. 2.3 Modifying Your First Java Program 42
4. 2.4 Displaying Text with printf 44
5. 2.5 Another Application: Adding Integers 45

1. 2.5.1 import Declarations 46
2. 2.5.2 Declaring and Creating a Scanner to Obtain User Input from the Keyboard 46
3. 2.5.3 Prompting the User for Input 47
4. 2.5.4 Declaring a Variable to Store an Integer and Obtaining an Integer from the Keyboard 47
5. 2.5.5 Obtaining a Second Integer 48
6. 2.5.6 Using Variables in a Calculation 48
7. 2.5.7 Displaying the Calculation Result 48
8. 2.5.8 Java API Documentation 49
9. 2.5.9 Declaring and Initializing Variables in Separate Statements 49

6. 2.6 Memory Concepts 49
7. 2.7 Arithmetic 50

8. 2.8 Decision Making: Equality and Relational Operators 54 9. 2.9 Wrap-Up 57

3. 3 Introduction to Classes, Objects, Methods and Strings 68

1. 3.1 Introduction 69

2. 3.2 Instance Variables, *set* Methods and *get* Methods 70

1. 3.2.1 Account Class with an Instance Variable, and *set* and *get* Methods 70

2. 3.2.2 AccountTest Class That Creates and Uses an Object of Class Account 73

3. 3.2.3 Compiling and Executing an App with Multiple Classes 76

4. 3.2.4 Account UML Class Diagram 76

5. 3.2.5 Additional Notes on Class AccountTest 78

6. 3.2.6 Software Engineering with private Instance Variables and public *set* and *get* Methods 78

3. 3.3 Account Class: Initializing Objects with Constructors 79

1. 3.3.1 Declaring an Account Constructor for Custom Object Initialization 80

2. 3.3.2 Class AccountTest: Initializing Account Objects When They're Created 81

4. 3.4 Account Class with a Balance; Floating-Point Numbers 82

1. 3.4.1 Account Class with a balance Instance Variable of Type double 83

2. 3.4.2 AccountTest Class to Use Class Account 85

5. 3.5 Primitive Types vs. Reference Types 88

6. 3.6 (Optional) GUI and Graphics Case Study: A Simple GUI 88 1. 3.6.1 What Is a Graphical User Interface? 90

2. 3.6.2 JavaFX Scene Builder and FXML 90

3. 3.6.3 **Welcome** App—Displaying Text and an Image 90

4. 3.6.4 Opening Scene Builder and Creating the File Welcome.fxml 90
5. 3.6.5 Adding an Image to the Folder Containing Welcome.fxml 92
6. 3.6.6 Creating a VBox Layout Container 92
7. 3.6.7 Configuring the VBox 92
8. 3.6.8 Adding and Configuring a Label 92
9. 3.6.9 Adding and Configuring an ImageView 94
10. 3.6.10 Previewing the **Welcome** GUI 95

7. 3.7 Wrap-Up 96

4. 4 Control Statements: Part 1; Assignment, ++ and -- Operators 104

1. 4.1 Introduction 105
2. 4.2 Algorithms 105
3. 4.3 Pseudocode 106
4. 4.4 Control Structures 106

1. 4.4.1 Sequence Structure in Java 107
2. 4.4.2 Selection Statements in Java 108
3. 4.4.3 Iteration Statements in Java 108
4. 4.4.4 Summary of Control Statements in Java 108

5. 4.5 if Single-Selection Statement 109
6. 4.6 if…else Double-Selection Statement 110

1. 4.6.1 Nested if…else Statements 111
2. 4.6.2 Dangling-else Problem 112
3. 4.6.3 Blocks 112
4. 4.6.4 Conditional Operator (?:) 113

7. 4.7 Student Class: Nested if…else Statements 113
8. 4.8 while Iteration Statement 116

9. 4.9 Formulating Algorithms: Counter-Controlled Iteration 118 10. 4.10 Formulating Algorithms: Sentinel-Controlled Iteration 122 11. 4.11 Formulating Algorithms: Nested Control Statements 129 12. 4.12 Compound Assignment Operators 133

13. 4.13 Increment and Decrement Operators 134

14. 4.14 Primitive Types 137

15. 4.15 (Optional) GUI and Graphics Case Study: Event Handling; Drawing Lines 138

    1. 4.15.1 Test-Driving the Completed **Draw Lines** App 138
    2. 4.15.2 Building the App's GUI 139
    3. 4.15.3 Preparing to Interact with the GUI Programmatically 143
    4. 4.15.4 Class DrawLinesController 145
    5. 4.15.5 Class DrawLines—The Main Application Class 147

16. 4.16 Wrap-Up 149

5. 5 Control Statements: Part 2; Logical Operators 164

    1. 5.1 Introduction 165
    2. 5.2 Essentials of Counter-Controlled Iteration 165
    3. 5.3 for Iteration Statement 166
    4. 5.4 Examples Using the for Statement 170
        1. 5.4.1 Application: Summing the Even Integers from 2 to 20 171
        2. 5.4.2 Application: Compound-Interest Calculations 172
    5. 5.5 do…while Iteration Statement 175
    6. 5.6 switch Multiple-Selection Statement 176
    7. 5.7 Class AutoPolicy Case Study: Strings in switch Statements 182
    8. 5.8 break and continue Statements 185

1. 5.8.1 break Statement 185
2. 5.8.2 continue Statement 186

9. 5.9 Logical Operators 187
   1. 5.9.1 Conditional AND (&&) Operator 187
   2. 5.9.2 Conditional OR (||) Operator 188
   3. 5.9.3 Short-Circuit Evaluation of Complex Conditions 189
   4. 5.9.4 Boolean Logical AND (&) and Boolean Logical Inclusive OR (|) Operators 189
   5. 5.9.5 Boolean Logical Exclusive OR (^) 190
   6. 5.9.6 Logical Negation (!) Operator 190
   7. 5.9.7 Logical Operators Example 191

10. 5.10 Structured-Programming Summary 193
11. 5.11 (Optional) GUI and Graphics Case Study: Drawing Rectangles and Ovals 198
12. 5.12 Wrap-Up 201

6. 6 Methods: A Deeper Look 212
   1. 6.1 Introduction 213
   2. 6.2 Program Units in Java 213
   3. 6.3 static Methods, static Fields and Class Math 215 4.

   6.4 Methods with Multiple Parameters 217
   5. 6.5 Notes on Declaring and Using Methods 221
   6. 6.6 Method-Call Stack and Activation Records 222
      1. 6.6.1 Method-Call Stack 222
      2. 6.6.2 Stack Frames 222
      3. 6.6.3 Local Variables and Stack Frames 222
      4. 6.6.4 Stack Overflow 223

   7. 6.7 Argument Promotion and Casting 223

8. 6.8 Java API Packages 224

9. 6.9 Case Study: Secure Random-Number Generation 226

10. 6.10 Case Study: A Game of Chance; Introducing enum Types 231

11. 6.11 Scope of Declarations 236

12. 6.12 Method Overloading 238

    1. 6.12.1 Declaring Overloaded Methods 238

    2. 6.12.2 Distinguishing Between Overloaded Methods 239

    3. 6.12.3 Return Types of Overloaded Methods 240

13. 6.13 (Optional) GUI and Graphics Case Study: Colors and Filled Shapes 240

14. 6.14 Wrap-Up 243

7. 7 Arrays and ArrayLists 257

    1. 7.1 Introduction 258

    2. 7.2 Arrays 259

    3. 7.3 Declaring and Creating Arrays 260

    4. 7.4 Examples Using Arrays 262

        1. 7.4.1 Creating and Initializing an Array 262

        2. 7.4.2 Using an Array Initializer 263

        3. 7.4.3 Calculating the Values to Store in an Array

        264 4. 7.4.4 Summing the Elements of an Array

        265

        5. 7.4.5 Using Bar Charts to Display Array Data Graphically 265

        6. 7.4.6 Using the Elements of an Array as

        Counters 267 7. 7.4.7 Using Arrays to Analyze

        Survey Results 268

    5. 7.5 Exception Handling: Processing the Incorrect Response 270

1. 7.5.1 The try Statement 270
2. 7.5.2 Executing the catch Block 270
3. 7.5.3 toString Method of the Exception Parameter 271

6. 7.6 Case Study: Card Shuffling and Dealing Simulation

271 7. 7.7 Enhanced for Statement 276

8. 7.8 Passing Arrays to Methods 277
9. 7.9 Pass-By-Value vs. Pass-By-Reference 279
10. 7.10 Case Study: Class GradeBook Using an Array to Store Grades 280
11. 7.11 Multidimensional Arrays 285
    1. 7.11.1 Arrays of One-Dimensional Arrays 286
    2. 7.11.2 Two-Dimensional Arrays with Rows of Different Lengths 286
    3. 7.11.3 Creating Two-Dimensional Arrays with Array Creation Expressions 287
    4. 7.11.4 Two-Dimensional Array Example: Displaying Element Values 287
    5. 7.11.5 Common Multidimensional-Array Manipulations Performed with for Statements 288
12. 7.12 Case Study: Class GradeBook Using a Two-Dimensional Array 289
13. 7.13 Variable-Length Argument Lists 295
14. 7.14 Using Command-Line Arguments 296
15. 7.15 Class Arrays 298
16. 7.16 Introduction to Collections and Class ArrayList 301
17. 7.17 (Optional) GUI and Graphics Case Study: Drawing Arcs 305
18. 7.18 Wrap-Up 308

8. 8 Classes and Objects: A Deeper Look 329

1. 8.1 Introduction 330
2. 8.2 Time Class Case Study 330
3. 8.3 Controlling Access to Members 335
4. 8.4 Referring to the Current Object’s Members with the this Reference 336
5. 8.5 Time Class Case Study: Overloaded

Constructors 338 6. 8.6 Default and No-Argument

Constructors 343

7. 8.7 Notes on *Set* and *Get* Methods 344
8. 8.8 Composition 345
9. 8.9 enum Types 348
10. 8.10 Garbage Collection 351
11. 8.11 static Class Members 351
12. 8.12 static Import 355
13. 8.13 final Instance Variables 356
14. 8.14 Package Access 357
15. 8.15 Using BigDecimal for Precise Monetary Calculations 358
16. 8.16 (Optional) GUI and Graphics Case Study: Using Objects with Graphics 361
17. 8.17 Wrap-Up 365

9. 9 Object-Oriented Programming: Inheritance 373

    1. 9.1 Introduction 374
    2. 9.2 Superclasses and Subclasses 375
    3. 9.3 protected Members 377
    4. 9.4 Relationship Between Superclasses and Subclasses 378
        1. 9.4.1 Creating and Using a CommissionEmployee Class 378
        2. 9.4.2 Creating and Using a BasePlusCommissionEmployee Class 383
        3. 9.4.3 Creating a

CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy 388

4. 9.4.4 CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using protected Instance Variables 391
5. 9.4.5 CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using private Instance Variables 394

5. 9.5 Constructors in Subclasses 398
6. 9.6 Class Object 399
7. 9.7 Designing with Composition vs. Inheritance 400
8. 9.8 Wrap-Up 402

10. 10 Object-Oriented Programming: Polymorphism and Interfaces 407
    1. 10.1 Introduction 408
    2. 10.2 Polymorphism Examples 410
    3. 10.3 Demonstrating Polymorphic Behavior 411
    4. 10.4 Abstract Classes and Methods 413
    5. 10.5 Case Study: Payroll System Using Polymorphism 416
        1. 10.5.1 Abstract Superclass Employee 417
        2. 10.5.2 Concrete Subclass SalariedEmployee 419
        3. 10.5.3 Concrete Subclass HourlyEmployee 421
        4. 10.5.4 Concrete Subclass CommissionEmployee 422
        5. 10.5.5 Indirect Concrete Subclass BasePlusCommissionEmployee 424
        6. 10.5.6 Polymorphic Processing, Operator instanceof and Downcasting 425
    6. 10.6 Allowed Assignments Between Superclass and Subclass Variables 430
    7. 10.7 final Methods and Classes 430
    8. 10.8 A Deeper Explanation of Issues with Calling Methods

from Constructors 431

9. 10.9 Creating and Using Interfaces 432
    1. 10.9.1 Developing a Payable Hierarchy 434
    2. 10.9.2 Interface Payable 435
    3. 10.9.3 Class Invoice 435
    4. 10.9.4 Modifying Class Employee to Implement Interface Payable 437
    5. 10.9.5 Using Interface Payable to Process Invoices and Employees Polymorphically 439
    6. 10.9.6 Some Common Interfaces of the Java API 440
10. 10.10 Java SE 8 Interface Enhancements 441
    1. 10.10.1 default Interface Methods 441
    2. 10.10.2 static Interface Methods 442
    3. 10.10.3 Functional Interfaces 442
11. 10.11 Java SE 9 private Interface Methods 443
12. 10.12 private Constructors 443
13. 10.13 Program to an Interface, Not an Implementation 444
    1. 10.13.1 Implementation Inheritance Is Best for Small Numbers of Tightly Coupled Classes 444
    2. 10.13.2 Interface Inheritance Is Best for Flexibility 444
    3. 10.13.3 Rethinking the Employee Hierarchy 445
14. 10.14 (Optional) GUI and Graphics Case Study: Drawing with Polymorphism 446
15. 10.15 Wrap-Up 448

11. 11 Exception Handling: A Deeper Look 455
    1. 11.1 Introduction 456
    2. 11.2 Example: Divide by Zero without Exception Handling

457 3. 11.3 Example: Handling ArithmeticExceptions and

InputMismatchExceptions 459

4. 11.4 When to Use Exception Handling 465
5. 11.5 Java Exception Hierarchy 465
6. 11.6 finally Block 469
7. 11.7 Stack Unwinding and Obtaining Information from an Exception 473
8. 11.8 Chained Exceptions 476
9. 11.9 Declaring New Exception Types 478
10. 11.10 Preconditions and Postconditions 479
11. 11.11 Assertions 479
12. 11.12 try-with-Resources: Automatic Resource Deallocation 481
13. 11.13 Wrap-Up 482

12. 12 JavaFX Graphical User Interfaces: Part 1 488
    1. 12.1 Introduction 489
    2. 12.2 JavaFX Scene Builder 490
    3. 12.3 JavaFX App Window Structure 491
    4. 12.4 **Welcome** App—Displaying Text and an Image 492
        1. 12.4.1 Opening Scene Builder and Creating the File Welcome.fxml 492
        2. 12.4.2 Adding an Image to the Folder Containing Welcome.fxml 493
        3. 12.4.3 Creating a VBox Layout Container 493
        4. 12.4.4 Configuring the VBox Layout Container 494
        5. 12.4.5 Adding and Configuring a Label 494
        6. 12.4.6 Adding and Configuring an ImageView 495
        7. 12.4.7 Previewing the Welcome GUI 497
    5. 12.5 **Tip Calculator** App—Introduction to Event Handling 497
        1. 12.5.1 Test-Driving the **Tip Calculator** App 498
        2. 12.5.2 Technologies Overview 499
        3. 12.5.3 Building the App’s GUI 501

4. 12.5.4 TipCalculator Class 508
5. 12.5.5 TipCalculatorController Class 510

6. 12.6 Features Covered in the Other JavaFX Chapters 515
7. 12.7 Wrap-Up 515

13. 13 JavaFX GUI: Part 2 523

1. 13.1 Introduction 524
2. 13.2 Laying Out Nodes in a Scene Graph 524
3. 13.3 **Painter** App: RadioButtons, Mouse Events and Shapes 526
    1. 13.3.1 Technologies Overview 526
    2. 13.3.2 Creating the Painter.fxml File 528
    3. 13.3.3 Building the GUI 528
    4. 13.3.4 Painter Subclass of Application 531
    5. 13.3.5 PainterController Class 532
4. 13.4 **Color Chooser** App: Property Bindings and Property Listeners 536
    1. 13.4.1 Technologies Overview 536
    2. 13.4.2 Building the GUI 537
    3. 13.4.3 ColorChooser Subclass of Application 539
    4. 13.4.4 ColorChooserController Class 540
5. 13.5 **Cover Viewer** App: Data-Driven GUIs with JavaFX Collections 542
    1. 13.5.1 Technologies Overview 543
    2. 13.5.2 Adding Images to the App’s Folder 543
    3. 13.5.3 Building the GUI 543
    4. 13.5.4 CoverViewer Subclass of Application 545
    5. 13.5.5 CoverViewerController Class 545

6. 13.6 **Cover Viewer** App: Customizing ListView Cells 547
    1. 13.6.1 Technologies Overview 548
    2. 13.6.2 Copying the CoverViewer App 548
    3. 13.6.3 ImageTextCell Custom Cell Factory Class 549
    4. 13.6.4 CoverViewerController Class 550
7. 13.7 Additional JavaFX Capabilities 551
8. 13.8 JavaFX 9: Java SE 9 JavaFX Updates 553
9. 13.9 Wrap-Up 555

14. 14 Strings, Characters and Regular Expressions 564
    1. 14.1 Introduction 565
    2. 14.2 Fundamentals of Characters and Strings 565
    3. 14.3 Class String 566
        1. 14.3.1 String Constructors 566
        2. 14.3.2 String Methods length, charAt and getChars 567
        3. 14.3.3 Comparing Strings 569
        4. 14.3.4 Locating Characters and Substrings in Strings 573
        5. 14.3.5 Extracting Substrings from Strings 575
        6. 14.3.6 Concatenating Strings 576
        7. 14.3.7 Miscellaneous String Methods 577
        8. 14.3.8 String Method valueOf 578
    4. 14.4 Class StringBuilder 579
        1. 14.4.1 StringBuilder Constructors 580
        2. 14.4.2 StringBuilder Methods length, capacity, setLength and ensureCapacity 581

3. 14.4.3 StringBuilder Methods charAt, setCharAt, getChars and reverse 582

4. 14.4.4 StringBuilder append Methods 583

5. 14.4.5 StringBuilder Insertion and Deletion Methods 585

5. 14.5 Class Character 586

6. 14.6 Tokenizing Strings 591

7. 14.7 Regular Expressions, Class Pattern and Class Matcher 592

1. 14.7.1 Replacing Substrings and Splitting Strings 597

2. 14.7.2 Classes Pattern and Matcher 599

8. 14.8 Wrap-Up 601

15. 15 Files, Input/Output Streams, NIO and XML Serialization 612

1. 15.1 Introduction 613

2. 15.2 Files and Streams 613
3. 15.3 Using NIO Classes and Interfaces to Get File and Directory Information 615

4. 15.4 Sequential Text Files 619

1. 15.4.1 Creating a Sequential Text File 619

2. 15.4.2 Reading Data from a Sequential Text File 622

3. 15.4.3 Case Study: A Credit-Inquiry Program 623

4. 15.4.4 Updating Sequential Files 628

5. 15.5 XML Serialization 628

1. 15.5.1 Creating a Sequential File Using XML Serialization 628

2. 15.5.2 Reading and Deserializing Data from a Sequential File 634

6. 15.6 FileChooser and DirectoryChooser Dialogs 635 7.

15.7 (Optional) Additional java.io Classes 641

1. 15.7.1 Interfaces and Classes for Byte-Based Input and Output 641
2. 15.7.2 Interfaces and Classes for Character-Based Input and Output 643

8. 15.8 Wrap-Up 644

16. 16 Generic Collections 652

1. 16.1 Introduction 653
2. 16.2 Collections Overview 653
3. 16.3 Type-Wrapper Classes 655
4. 16.4 Autoboxing and Auto-Unboxing 655
5. 16.5 Interface Collection and Class Collections 655
6. 16.6 Lists 656
   1. 16.6.1 ArrayList and Iterator 657
   2. 16.6.2 LinkedList 659
7. 16.7 Collections Methods 664
   1. 16.7.1 Method sort 664
   2. 16.7.2 Method shuffle 668
   3. 16.7.3 Methods reverse, fill, copy, max and min 670
   4. 16.7.4 Method binarySearch 672
   5. 16.7.5 Methods addAll, frequency and disjoint 673
8. 16.8 Class PriorityQueue and Interface Queue 675
9. 16.9 Sets 676
10. 16.10 Maps 679
11. 16.11 Synchronized Collections 683
12. 16.12 Unmodifiable Collections 683
13. 16.13 Abstract Implementations 684

14. 16.14 Java SE 9: Convenience Factory Methods for Immutable Collections 684

15. 16.15 Wrap-Up 688

17. 17 Lambdas and Streams 694

1. 17.1 Introduction 695
2. 17.2 Streams and Reduction 697

1. 17.2.1 Summing the Integers from 1 through 10 with a for Loop 697
2. 17.2.2 External Iteration with for Is Error Prone

698 3. 17.2.3 Summing with a Stream and

Reduction 698 4. 17.2.4 Internal Iteration 699

3. 17.3 Mapping and Lambdas 700

1. 17.3.1 Lambda Expressions 701
2. 17.3.2 Lambda Syntax 702
3. 17.3.3 Intermediate and Terminal Operations 703

4. 17.4 Filtering 704
5. 17.5 How Elements Move Through Stream

Pipelines 706 6. 17.6 Method References 707

1. 17.6.1 Creating an IntStream of Random Values 708
2. 17.6.2 Performing a Task on Each Stream Element with forEach and a Method Reference 708
3. 17.6.3 Mapping Integers to String Objects with mapToObj 709
4. 17.6.4 Concatenating Strings with collect 709

7. 17.7 IntStream Operations 710

1. 17.7.1 Creating an IntStream and Displaying Its Values 711
2. 17.7.2 Terminal Operations count, min, max,

sum and average 711

3. 17.7.3 Terminal Operation reduce 712
4. 17.7.4 Sorting IntStream Values 714

8. 17.8 Functional Interfaces 715
9. 17.9 Lambdas: A Deeper Look 716
10. 17.10 Stream<Integer> Manipulations 717
    1. 17.10.1 Creating a Stream<Integer> 718
    2. 17.10.2 Sorting a Stream and Collecting the Results 719
    3. 17.10.3 Filtering a Stream and Storing the Results for Later Use 719
    4. 17.10.4 Filtering and Sorting a Stream and Collecting the Results 720
    5. 17.10.5 Sorting Previously Collected Results 720
11. 17.11 Stream<String> Manipulations 720
    1. 17.11.1 Mapping Strings to Uppercase 721
    2. 17.11.2 Filtering Strings Then Sorting Them in Case Insensitive Ascending Order 722
    3. 17.11.3 Filtering Strings Then Sorting Them in Case Insensitive Descending Order 722
12. 17.12 Stream<Employee> Manipulations 723
    1. 17.12.1 Creating and Displaying a List<Employee> 724
    2. 17.12.2 Filtering Employees with Salaries in a Specified Range 725
    3. 17.12.3 Sorting Employees By Multiple Fields 728
    4. 17.12.4 Mapping Employees to Unique-Last-Name Strings 730
    5. 17.12.5 Grouping Employees By Department 731
    6. 17.12.6 Counting the Number of Employees in Each Department 732

7. 17.12.7 Summing and Averaging Employee Salaries 733
13. 17.13 Creating a Stream<String> from a File 734
14. 17.14 Streams of Random Values 737
15. 17.15 Infinite Streams 739
16. 17.16 Lambda Event Handlers 741
17. 17.17 Additional Notes on Java SE 8 Interfaces 741
18. 17.18 Wrap-Up 742

18. 18 Recursion 756
    1. 18.1 Introduction 757
    2. 18.2 Recursion Concepts 758
    3. 18.3 Example Using Recursion: Factorials 759
    4. 18.4 Reimplementing Class FactorialCalculator Using BigInteger 761
    5. 18.5 Example Using Recursion: Fibonacci Series 763
    6. 18.6 Recursion and the Method-Call Stack 766
    7. 18.7 Recursion vs. Iteration 767
    8. 18.8 Towers of Hanoi 769
    9. 18.9 Fractals 771
        1. 18.9.1 Koch Curve Fractal 772
        2. 18.9.2 (Optional) Case Study: Lo Feather Fractal 773
        3. 18.9.3 (Optional) **Fractal** App GUI 775
        4. 18.9.4 (Optional) FractalController Class 777
    10. 18.10 Recursive Backtracking 782
    11. 18.11 Wrap-Up 782

19. 19 Searching, Sorting and Big O 791
    1. 19.1 Introduction 792
    2. 19.2 Linear Search 793
    3. 19.3 Big O Notation 796

1. 19.3.1 *O*(1) Algorithms 796
2. 19.3.2 *O*(*n*) Algorithms 796
3. 19.3.3 *O*(n^2) Algorithms 796
4. 19.3.4 Big O of the Linear Search 797

4. 19.4 Binary Search 797
   1. 19.4.1 Binary Search Implementation 798
   2. 19.4.2 Efficiency of the Binary Search 801
5. 19.5 Sorting Algorithms 802
6. 19.6 Selection Sort 802
   1. 19.6.1 Selection Sort Implementation 803
   2. 19.6.2 Efficiency of the Selection Sort 805
7. 19.7 Insertion Sort 805
   1. 19.7.1 Insertion Sort Implementation 806
   2. 19.7.2 Efficiency of the Insertion Sort 808
8. 19.8 Merge Sort 809
   1. 19.8.1 Merge Sort Implementation 809
   2. 19.8.2 Efficiency of the Merge Sort 814
9. 19.9 Big O Summary for This Chapter's Searching and Sorting Algorithms 814
10. 19.10 Massive Parallelism and Parallel Algorithms 815
11. 19.11 Wrap-Up 815

20. 20 Generic Classes and Methods: A Deeper Look 821
    1. 20.1 Introduction 822
    2. 20.2 Motivation for Generic Methods 822
    3. 20.3 Generic Methods: Implementation and Compile-Time Translation 824

4. 20.4 Additional Compile-Time Translation Issues: Methods That Use a Type Parameter as the Return Type 827

5. 20.5 Overloading Generic Methods 830

6. 20.6 Generic Classes 831

7. 20.7 Wildcards in Methods That Accept Type Parameters 838

8. 20.8 Wrap-Up 842

21. 21 Custom Generic Data Structures 846

1. 21.1 Introduction 847

2. 21.2 Self-Referential Classes 848

3. 21.3 Dynamic Memory Allocation 848

4. 21.4 Linked Lists 849

1. 21.4.1 Singly Linked Lists 849

2. 21.4.2 Implementing a Generic List Class 850

3. 21.4.3 Generic Classes ListNode and List 853

4. 21.4.4 Class ListTest 853

5. 21.4.5 List Method insertAtFront 855

6. 21.4.6 List Method insertAtBack 856

7. 21.4.7 List Method removeFromFront 856

8. 21.4.8 List Method removeFromBack 857

9. 21.4.9 List Method print 858

10. 21.4.10 Creating Your Own Packages 858

5. 21.5 Stacks 863

6. 21.6 Queues 866

7. 21.7 Trees 868

8. 21.8 Wrap-Up 875

22. 22 JavaFX Graphics and Multimedia 900

1. 22.1 Introduction 901

2. 22.2 Controlling Fonts with Cascading Style Sheets (CSS) 902

1. 22.2.1 CSS That Styles the GUI 902
2. 22.2.2 FXML That Defines the GUI—Introduction to XML Markup 905
3. 22.2.3 Referencing the CSS File from FXML 908
4. 22.2.4 Specifying the VBox's Style Class 908
5. 22.2.5 Programmatically Loading CSS 908

3. 22.3 Displaying Two-Dimensional Shapes 909

1. 22.3.1 Defining Two-Dimensional Shapes with FXML 909
2. 22.3.2 CSS That Styles the Two-Dimensional Shapes 912

4. 22.4 Polylines, Polygons and Paths 914

1. 22.4.1 GUI and CSS 915
2. 22.4.2 PolyShapesController Class 916

5. 22.5 Transforms 919
6. 22.6 Playing Video with Media, MediaPlayer and MediaViewer 921

1. 22.6.1 VideoPlayer GUI 922
2. 22.6.2 VideoPlayerController Class 924

7. 22.7 Transition Animations 928

1. 22.7.1 TransitionAnimations.fxml 928
2. 22.7.2 TransitionAnimationsController Class 930

8. 22.8 Timeline Animations 934
9. 22.9 Frame-by-Frame Animation with AnimationTimer 937
10. 22.10 Drawing on a Canvas 939
11. 22.11 Three-Dimensional Shapes 944
12. 22.12 Wrap-Up 947

23. 23 Concurrency 963

    1. 23.1 Introduction 964
    2. 23.2 Thread States and Life Cycle 966
        1. 23.2.1 *New* and *Runnable* States 967
        2. 23.2.2 *Waiting* State 967
        3. 23.2.3 *Timed Waiting* State 967
        4. 23.2.4 *Blocked* State 967
        5. 23.2.5 *Terminated* State 967
        6. 23.2.6 Operating-System View of the *Runnable* State 968
        7. 23.2.7 Thread Priorities and Thread Scheduling 968
        8. 23.2.8 Indefinite Postponement and Deadlock 969
    3. 23.3 Creating and Executing Threads with the Executor Framework 969
    4. 23.4 Thread Synchronization 973
        1. 23.4.1 Immutable Data 974
        2. 23.4.2 Monitors 974
        3. 23.4.3 Unsynchronized Mutable Data Sharing 975
        4. 23.4.4 Synchronized Mutable Data Sharing—Making Operations Atomic 979
    5. 23.5 Producer/Consumer Relationship without Synchronization 982
    6. 23.6 Producer/Consumer Relationship: ArrayBlockingQueue 990
    7. 23.7 (Advanced) Producer/Consumer Relationship with synchronized, wait, notify and notifyAll 993
    8. 23.8 (Advanced) Producer/Consumer Relationship: Bounded Buffers 999
    9. 23.9 (Advanced) Producer/Consumer Relationship: The Lock and Condition Interfaces 1007
    10. 23.10 Concurrent Collections 1014

11. 23.11 Multithreading in JavaFX 1016
    1. 23.11.1 Performing Computations in a Worker Thread: Fibonacci Numbers 1017
    2. 23.11.2 Processing Intermediate Results: Sieve of Eratosthenes 1022
12. 23.12 sort/parallelSort Timings with the Java SE 8 Date/Time API 1028
13. 23.13 Java SE 8: Sequential vs. Parallel Streams 1031
14. 23.14 (Advanced) Interfaces Callable and Future 1033
15. 23.15 (Advanced) Fork/Join Framework 1038
16. 23.16 Wrap-Up 1038

24. 24 Accessing Databases with JDBC 1050
    1. 24.1 Introduction 1051
    2. 24.2 Relational Databases 1052
    3. 24.3 A books Database 1053
    4. 24.4 SQL 1057
        1. 24.4.1 Basic SELECT Query 1058
        2. 24.4.2 WHERE Clause 1058
        3. 24.4.3 ORDER BY Clause 1060
    5. 24.4.4 Merging Data from Multiple Tables: INNER JOIN 1062
        1. 24.4.5 INSERT Statement 1063
        2. 24.4.6 UPDATE Statement 1064
        3. 24.4.7 DELETE Statement 1065
    6. 24.5 Setting Up a Java DB Database 1066
        1. 24.5.1 Creating the Chapter's Databases on Windows 1067
        2. 24.5.2 Creating the Chapter's Databases on macOS

1068

3. 24.5.3 Creating the Chapter's Databases on Linux 1068

7. 24.6 Connecting to and Querying a Database 1068

1. 24.6.1 Automatic Driver Discovery 1070
2. 24.6.2 Connecting to the Database 1070
3. 24.6.3 Creating a Statement for Executing Queries 1071
4. 24.6.4 Executing a Query 1071
5. 24.6.5 Processing a Query's ResultSet 1072

8. 24.7 Querying the books Database 1073

1. 24.7.1 ResultSetTableModel Class 1073
2. 24.7.2 DisplayQueryResults App's GUI 1080
3. 24.7.3 DisplayQueryResultsController Class 1080

9. 24.8 RowSet Interface 1085
10. 24.9 PreparedStatements 1088

1. 24.9.1 AddressBook App That Uses PreparedStatements 1089
2. 24.9.2 Class Person 1089
3. 24.9.3 Class PersonQueries 1091
4. 24.9.4 AddressBook GUI 1094
5. 24.9.5 Class AddressBookController 1095

11. 24.10 Stored Procedures 1100
12. 24.11 Transaction Processing 1100
13. 24.12 Wrap-Up 1101

25. 25 Introduction to JShell: Java 9's REPL 1109

1. 25.1 Introduction 1110
2. 25.2 Installing JDK 9 1112

3. 25.3 Introduction to JShell 1112
    1. 25.3.1 Starting a JShell Session 1113
    2. 25.3.2 Executing Statements 1113
    3. 25.3.3 Declaring Variables Explicitly 1114
    4. 25.3.4 Listing and Executing Prior Snippets 1116
    5. 25.3.5 Evaluating Expressions and Declaring Variables Implicitly 1118
    6. 25.3.6 Using Implicitly Declared Variables

1118 7. 25.3.7 Viewing a Variable's Value 1119

    8. 25.3.8 Resetting a JShell Session 1119
    9. 25.3.9 Writing Multiline Statements 1119
    10. 25.3.10 Editing Code Snippets 1120
    11. 25.3.11 Exiting JShell 1123

4. 25.4 Command-Line Input in JShell 1123
5. 25.5 Declaring and Using Classes 1124
    1. 25.5.1 Creating a Class in JShell 1125
    2. 25.5.2 Explicitly Declaring Reference-Type Variables 1125
    3. 25.5.3 Creating Objects 1126
    4. 25.5.4 Manipulating Objects 1126
    5. 25.5.5 Creating a Meaningful Variable Name for an Expression 1127
    6. 25.5.6 Saving and Opening Code-Snippet Files 1128

6. 25.6 Discovery with JShell Auto-Completion 1128
    1. 25.6.1 Auto-Completing Identifiers 1129
    2. 25.6.2 Auto-Completing JShell Commands 1130 7. 25.7

Exploring a Class's Members and Viewing Documentation

1130

1. 25.7.1 Listing Class Math's static Members 1131
2. 25.7.2 Viewing a Method's Parameters 1131
3. 25.7.3 Viewing a Method's Documentation 1132 4.

25.7.4 Viewing a public Field's Documentation 1132

5. 25.7.5 Viewing a Class's Documentation 1133
6. 25.7.6 Viewing Method Overloads 1133
7. 25.7.7 Exploring Members of a Specific Object 1134

8. 25.8 Declaring Methods 1136

1. 25.8.1 Forward Referencing an Undeclared Method— Declaring Method displayCubes 1136
2. 25.8.2 Declaring a Previously Undeclared Method 1136
3. 25.8.3 Testing cube and Replacing Its Declaration 1137
4. 25.8.4 Testing Updated Method cube and Method displayCubes 1137

9. 25.9 Exceptions 1138
10. 25.10 Importing Classes and Adding Packages to the CLASSPATH 1139
11. 25.11 Using an External Editor 1141
12. 25.12 Summary of JShell Commands 1143

1. 25.12.1 Getting Help in JShell 1144
2. 25.12.2 /edit Command: Additional Features

1145 3. 25.12.3 /reload Command 1145

4. 25.12.4 /drop Command 1146
5. 25.12.5 Feedback Modes 1146
6. 25.12.6 Other JShell Features Configurable with /set 1148

13. 25.13 Keyboard Shortcuts for Snippet Editing 1149
14. 25.14 How JShell Reinterprets Java for Interactive Use

1149 15. 25.15 IDE JShell Support 1150

16. 25.16 Wrap-Up 1150

1. Chapters on the Web 1166
2. A Operator Precedence Chart 1167
3. B ASCII Character Set 1169
4. C Keywords and Reserved Words 1170
5. D Primitive Types 1171
6. E Using the Debugger 1172
    1. E.1 Introduction 1173
    2. E.2 Breakpoints and the run, stop, cont and print Commands 1173
    3. E.3 The print and set Commands 1177
    4. E.4 Controlling Execution Using the step, step up and next Commands 1179
    5. E.5 The watch Command 1181
    6. E.6 The clear Command 1183
    7. E.7 Wrap-Up 1186
7. Appendices on the Web 1187
8. Index 1189

1. 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. 2. 26 Swing GUI Components: Part 1

3. 27 Graphics and Java 2D
4. 28 Networking
5. 29 Java Persistence API (JPA)
6. 30 JavaServer™ Faces Web Apps: Part 1
7. 31 JavaServer™ Faces Web Apps: Part 2
8. 32 REST-Based Web Services
9. 33 (Optional) ATM Case Study, Part 1: Object-Oriented Design

with the UML

10. 34 (Optional) ATM Case Study, Part 2: Implementing an Object-Oriented Design
11. 35 Swing GUI Components: Part 2
12. 36 Java Module System and Other Java 9 Features
13. F Using the Java API Documentation
14. G Creating Documentation with javadoc
15. H Unicode®
16. I Formatted Output
17. J Number Systems
18. K Bit Manipulation
19. L Labeled break and continue Statements
20. M UML 2: Additional Diagram Types
21. 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]

[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.)

Oracle Certified Associate, Java SE 8 Programmer

Oracle Certified Professional, Java SE 8 Programmer

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

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

New and Updated Features

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

Flexibility Using Java SE 8 or the New Java SE 9 (which includes Java SE 8)

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

Introduction and Programming Fundamentals

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

Object-Oriented Programming

Flexible JavaFX/Swing GUI, Graphics 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

Type-inference improvements

@FunctionalInterface annotation

Bulk data operations for Java Collections—filter, map and reduce

Library enhancements to support lambdas (e.g., java.util.stream,

Date & Time API (java.time)

Parallel array sorting

Java concurrency API improvements

static and default methods in interfaces

Functional interfaces that define only one abstract method and can include static and default

java.util.function) methods

Fig. 1

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

Java 9 features

On the Companion Website

Module system

HTML5 Javadoc enhancements

In the Print Book

New JShell chapter

_ is no longer allowed as an identifier

private interface methods

Effectively final variables can be used in try-with-resources statements

Mention of the Stack Walking API

Mention of JEP 254, Compact Strings

Collection factory methods

Matcher class's new method overloads

CompletableFuture enhancements

JavaFX 9 skin APIs and other

enhancements

Mentions of:

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]

[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.

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

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.

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)

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
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.

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

9

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	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

Drawing Application	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 GUI via 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.