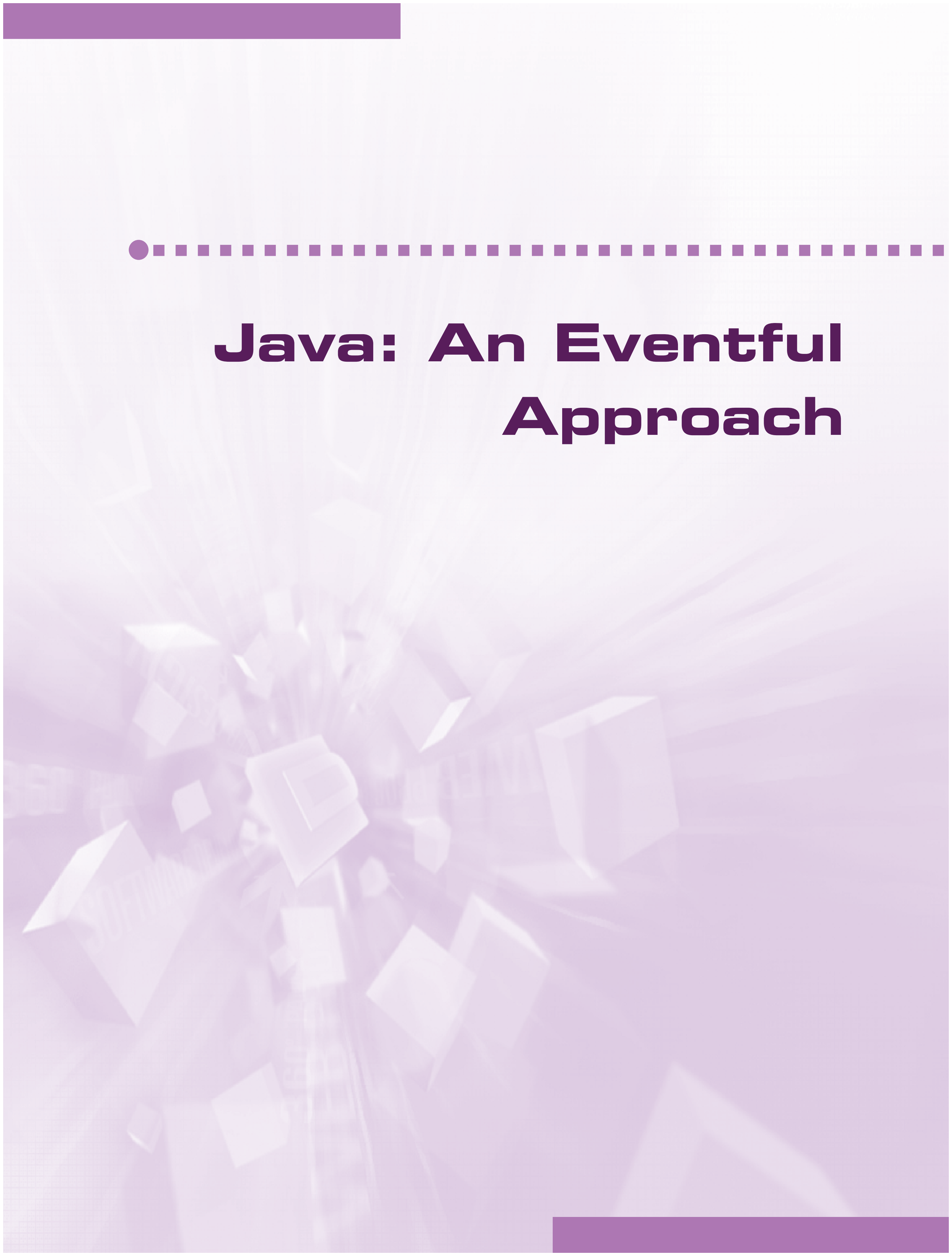

Java: An Eventful Approach

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

Java: An Eventful Approach

Kim B. Bruce
Pomona College

Andrea Pohoreckyj Danyluk
Thomas P. Murtagh
Williams College

PEARSON
Prentice
Hall

UPPER SADDLE RIVER, NEW JERSEY 07458

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

Library of Congress Cataloging-in-Publication Data
CIP Data Available

Vice President and Editorial Director, ECS: Marcia Horton
Senior Acquisitions Editor: Tracy Dunkelberger
Editorial Assistant: Christianna Lee
Executive Managing Editor: Vince O'Brien
Managing Editor: Camille Trentacoste
Production Editor: Irwin Zucker
Director of Creative Services: Paul Belfanti
Art Director: Maureen Eide
Cover Designer: Suzanne Behnke
Cover Illustration: Fireworks, Photodisc Collection / Getty Images, Inc.
Managing Editor, AV Management and Production: Patricia Burns
Art Editor: Xiaohong Zhu
Manufacturing Buyer: Lisa McDowell
Marketing Manager: Pamela Hersperger
Marketing Assistant: Barrie Reinhold

© 2006 by Pearson Education, Inc.
Pearson Prentice Hall
Pearson Education, Inc.
Upper Saddle River, NJ 07458

All rights reserved. No part of this book may be reproduced in any form or by any means, without permission in writing from the publisher.

Pearson Prentice Hall® is a trademark of Pearson Education, Inc.

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-146913-4

Pearson Education Ltd., London
Pearson Education Australia Pty. Ltd., Sydney
Pearson Education Singapore, Pte. Ltd.
Pearson Education North Asia Ltd., Hong Kong
Pearson Education Canada, Inc., Toronto
Pearson Educación de Mexico, S.A. de C.V.
Pearson Education-Japan, Tokyo
Pearson Education Malaysia, Pte. Ltd.
Pearson Education, Inc., Upper Saddle River, New Jersey

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

To my wife, Fatma. – Kim

To my children, Stephan and Katya, and my husband, Andrew. – Andrea

To my wife, Fern. – Tom

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

Contents

Preface xxix

0.1 Target Audience xxix
0.2 For the Student xxix
0.2.1 Mysterious Buzzwords xxix
0.2.2 How to Read This Book xxx
0.3 For the Instructor xxx
0.3.1 Special Features of This Text xxx
0.3.2 Why Introduce a Library? xxxii
0.3.3 Supplementary Materials for Instructors xxxiii
0.4 Flexibility for the Instructor and Student xxxiii
0.5 Additional Practical Information xxxiv
0.6 Acknowledgements xxxv

1 What Is Programming Anyway? 1

1.1 Without Understanding 2
1.2 The Java Programming Language 6
1.3 Your First Sip of Java 7
1.3.1 Simple Responsive Programs 8
1.3.2 "Class" and Other Magic Words 9
1.3.3 Discourse on the Method 11
1.4 Programming Tools 13
1.5 Drawing Primitives 17
1.5.1 The Graphics Coordinate System 17
1.5.2 Constructing Graphic Objects 19
1.6 Additional Event-Handling Methods 24
1.6.1 Mouse-Event-Handling Methods 24
1.6.2 The `begin` Method 25

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

1.7 To Err Is Human 26
1.8 Summary 29
1.9 Chapter Review Problems 30
1.10 Programming Problems 31

2 What's in a Name? 32

2.1 Naming and Modifying Objects 32
2.1.1 Mutator Methods 33
2.1.2 Instance Variable Declarations 35
2.1.3 Assigning Meanings to Variable Names 36
2.1.4 Comments 37
2.1.5 Additional Mutator Methods 38
2.1.6 Exercises 40
2.2 Nongraphical Classes of Objects 43
2.2.1 The Class of Colors 43
2.2.2 The Location Class 45
2.3 Layering on the Canvas 49
2.4 Accessing the Location of the Mouse 51
2.5 Sharing Parameter Information between Methods 53
2.6 Summary 58
2.7 Chapter Review Problems 59
2.8 Programming Problems 60

3 Working with Numbers 62

3.1 Introduction to Accessor Methods 62
3.2 Accessing Numerical Attributes of the Canvas 64
3.3 Expressions and Statements 66
3.4 Arithmetic Expressions 68
3.4.1 Ordering of Arithmetic Operations 71
3.5 Numeric Instance Variables 72
3.6 Initializers 75
3.7 Naming Numeric Constants 76
3.8 Displaying Numeric Information 78
3.8.1 Displaying Numbers as `Text` 78
3.8.2 Using `System.out.println` 80
3.8.3 Mixing Text and Numbers 83
3.9 Random Numbers 85
3.10 Summary 86
3.11 Chapter Review Problems 88
3.12 Programming Problems 89

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

4 Making Choices 92

4.1 A Brief Example: Using the `if` Statement to Count Votes 93
4.2 The `if` Statement 95
4.2.1 Example: Using the `if` Statement with 2-D Objects 97
4.3 Understanding Conditions 99
4.3.1 The `boolean` Data Type 99
4.4 Selecting Among Many Alternatives 102
4.5 More on Boolean Expressions 105
4.6 Nested Conditionals 109
4.7 Summary 115
4.8 Chapter Review Problems 115
4.9 Programming Problems 118

5 Primitive Types, Operators, and Strings 120

5.1 Operators vs. Method Invocations 121
5.2 Separate But Equal 126
5.3 Operators and Precedence Rules 128
5.4 Double Time 129
5.4.1 Java Needs More Than One Numeric Type 129
5.4.2 Arithmetic with `double`s and `int`s 131
5.4.3 Displaying `double`s 134
5.4.4 Why Are Rational Numbers Called `double`? 135
5.4.5 Selecting a Numeric Type 136
5.5 Handy Sources of Numeric Information 136
5.5.1 What Time Is It? 137
5.5.2 Sines and Wonders 139
5.6 Strings 143
5.7 Chapter Review Problems 146
5.8 Programming Problems 149

6 Classes 151

6.1 An Example Without Classes 151
6.2 Writing Classes: `FunnyFace` 155
6.2.1 Instance Variables 155
6.2.2 Methods and Parameters 156
6.2.3 Writing Accessor Methods 157
6.2.4 Constructors 159
6.2.5 Putting It All Together 161
6.3 Adding Methods to `FunnyFace` 163
6.3.1 Some Methods Are Like Those Already Written 164
6.3.2 Defining Methods Indirectly 164
6.3.3 Using `this` as a Parameter 166
6.4 Another Example: Implementing a `Timer` Class 167

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

6.5 Local Variables 169
6.6 Overloaded Methods and Constructors 172
6.7 Summary 177
6.8 Chapter Review Problems 177
6.9 Programming Problems 178

7 Control Structures 180

7.1 Repetition and `while` Loops 180
7.2 More Examples Using `while` Loops 185
7.3 Loops that Count 189
7.4 Nested Loops 190
7.5 Style Guidelines for Control Structures 192
7.6 DeMorgan’s Laws and Complex Boolean Expressions* 197
7.7 Simplifying Syntax in Conditionals 199
7.8 The `switch` Statement* 205
7.9 Summary 209
7.10 Chapter Review Problems 209
7.11 Programming Problems 210

8 Declarations and Scope 213

8.1 Access Control: `public` vs. `private` 213
8.2 Using Instance Variables, Parameters, and Local Variables 217
8.3 Scope of Identifiers 220
8.4 The Use of `static` 223
8.5 Summary 224
8.6 Chapter Review Problems 224

9 Active Objects 226

9.1 Animation 226
9.2 Active Objects 227
9.3 `Image`s and `VisibleImage`s 230
9.4 Interacting with Active Objects 232
9.5 Making Active Objects Affect Other Objects 234
9.5.1 Interacting with a Nonactive Object 235
9.5.2 Active Objects that Construct Other Active Objects 237
9.6 Active Objects without Loops 240
9.7 Making Animations Smooth 242
9.8 More Hints about Timing 244
9.9 Summary 245
9.10 Chapter Review Problems 246
9.11 Programming Problems 247

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

10 Interfaces 248

10.1 Interfaces for Flexibility 248
10.1.1 A First Example 248
10.1.2 Associating Objects and Classes With Interfaces 249
10.1.3 Using Interfaces 254
10.1.4 Other Features of Interfaces 257
10.1.5 Summary 257
10.2 Using Interfaces in the objectdraw Library* 258
10.3 Extending Interfaces* 260
10.4 Summary 263
10.5 Chapter Review Problems 264
10.6 Programming Problems 265

11 Graphical User Interfaces in Java 267

11.1 Text Fields 268
11.1.1 Constructing a Text Field 268
11.1.2 Adding a Text Field to a Window 270
11.1.3 Getting Information from a Text Field 272
11.2 Buttons and Events in Java 272
11.2.1 Creating and Adding Buttons 272
11.2.2 Handling Events 274
11.3 Checklist for Using GUI Components in a Program 275
11.4 Combo Boxes 276
11.5 Panels and Layout Managers 282
11.5.1 Panels 282
11.5.2 More on Layout Managers 286
11.6 Other GUI Components 288
11.6.1 Sliders 288
11.6.2 Labels 291
11.6.3 `JTextField` and `JTextArea` 292
11.7 Handling Keystroke and Mouse Events 295
11.7.1 Keystroke Events 295
11.7.2 Mouse Events 298
11.8 Summary 300
11.9 Chapter Review Problems 302
11.10 Programming Problems 303

12 Recursion 304

12.1 Recursive Structures 305
12.1.1 Nested Rectangles 305
12.1.2 Building and Searching Data Collections with Recursive Structures 315
12.1.3 Designing Recursive Structures 318

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

12.1.4 Why Does This Work? 321
12.1.5 Broccoli 322
12.2 Recursive Methods 328
12.2.1 Fast Exponentiation 329
12.2.2 Towers of Hanoi 332
12.3 Summary 335
12.4 Chapter Review Problems 336
12.5 Programming Problems 337

13 General Loops in Java 338

13.1 Recognizing Patterns with Loops 338
13.2 Counting and `for` Loops 339
13.2.1 Examples of Using `for` Loops 341
13.2.2 Other Variations on `for` Loops 344
13.3 The `do...while` Loop 345
13.4 Avoiding Loop Errors 347
13.5 Summary 350
13.6 Chapter Review Problems 351
13.7 Programming Problems 352

14 Arrays 354

14.1 Declaring Array Names 355
14.2 Creating an Array 356
14.2.1 Array Initializers 358
14.3 Using Arrays: A Triangle Class 359
14.4 Enhanced `for` loop in Java 1.5* 364
14.5 Gathering Information from an Array 365
14.5.1 Counting Speeders 367
14.5.2 Drawing a Histogram 368
14.6 Collections with Variable Sizes 373
14.6.1 Parallel Arrays vs. Arrays of Objects 376
14.6.2 Keeping Track of a Collection's Size 376
14.6.3 Adding an Entry to the Array 378
14.6.4 Displaying the Results 380
14.7 Adding and Removing Elements 386
14.7.1 Adding an Element to an Ordered Array 387
14.7.2 Removing an Element from an Array 392
14.8 Summary 396
14.9 Chapter Review Problems 397
14.10 Programming Problems 402

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

15 Multidimensional Arrays 406

15.1 General Two-Dimensional Arrays 407
15.1.1 Declaring an Array of Arrays 407
15.1.2 Creating an Array of Arrays 407
15.1.3 Indexing an Array of Arrays 410
15.1.4 Traversing a Two-Dimensional Array 411
15.1.5 Beyond Two Dimensions: Extending the Calendar Program 413
15.2 Matrices 417
15.2.1 Magic Squares 418
15.2.2 Declaring and Constructing a Matrix 419
15.2.3 Traversing a Matrix 419
15.2.4 Filling a Magic Square 422
15.2.5 Digital Image Smoothing 425
15.3 Summary 430
15.4 Chapter Review Problems 431
15.5 Programming Problems 433

16 Strings and Characters 436

16.1 Little Strings and Big Strings 437
16.1.1 The Empty String 437
16.1.2 Long Strings 437
16.2 A Collection of Useful `String` Methods 439
16.2.1 Building a String of URLs 440
16.2.2 Finding the Position of a Substring with `indexOf` 441
16.2.3 Dealing with Lower and Upper Case 442
16.2.4 Cutting and Pasting Strings 445
16.2.5 Trimming Strings 446
16.2.6 Comparing Strings 447
16.3 Characters 450
16.3.1 Characters and Strings 451
16.3.2 Performing Operations on Characters 454
16.4 Summary 457
16.5 Chapter Review Problems 458
16.6 Programming Problems 461

17 Inheritance 462

17.1 Extension as Specialization 462
17.2 Using `extends` 463
17.3 Protected vs. Public 470
17.4 Designing for Extension 472
17.5 Inheritance Hierarchies 477
17.5.1 Subclass Type Compatibility 479

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

17.6 Overriding Method Definitions 482
17.6.1 The `Object` class and the `equals` and `toString` Methods 483
17.6.2 Using `super` to Access Overridden Methods 483
17.6.3 Dynamic Method Invocation 486
17.6.4 Planned Obsolescence 488
17.6.5 Abstract Classes and Methods 493
17.7 Summary 493
17.8 Chapter Review Problems 494
17.9 Programming Problems 502

18 Exceptions 505

18.1 Exception Handlers 506
18.2 Exceptions Are Objects 510
18.3 Exception Propagation 510
18.4 Handling Multiple Exceptions 513
18.5 Exception Subclasses 514
18.6 Checked vs. Unchecked Exceptions 517
18.7 Throwing Exceptions 520
18.8 Summary 522
18.9 Chapter Review Problems 523
18.10 Programming Problems 525

19 Streams 527

19.1 Text Streams 528
19.1.1 Bookmark Files: An Example 528
19.1.2 Creating a `Writer` 530
19.1.3 Sending Data Through a `Writer` 531
19.1.4 `PrintWriter`s 535
19.1.5 Composing `Writer` classes 536
19.2 Readers 538
19.3 Writing an Application 543
19.4 Working with the File System 546
19.5 Sockets and Streams 552
19.5.1 Clients, and Servers 553
19.5.2 Networking Protocols 554
19.5.3 Creating and Using a `Socket` 556
19.5.4 Receiving Information from a Server 557
19.5.5 Extracting Information from a Web Page 561
19.6 Summary 564
19.7 Chapter Review Problems 565
19.8 Programming Problems 570

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

20 Searching and Sorting 573

20.1 Searching 573
20.1.1 Linear Searches 574
20.1.2 Recursive Processing of Arrays 574
20.1.3 Complexity of Linear Search 576
20.1.4 Binary Searches 577
20.2 Using Preconditions and Postconditions 582
20.3 Sorting 583
20.3.1 Selection Sort 583
20.3.2 Complexity of Selection Sort 588
20.3.3 Insertion Sort 588
20.3.4 Complexity of Insertion Sort 590
20.3.5 Merge Sort 592
20.3.6 Complexity of Merge Sort 596
20.4 Impact of the Complexity of Searching and Sorting 597
20.5 Summary 598
20.6 Chapter Review Problems 599

21 Introduction to Object-Oriented Design 600

21.1 What Is Object-Oriented Design? 600
21.2 Properties and Behaviors 601
21.2.1 Modeling Properties of Objects 601
21.2.2 Modeling Behavior 602
21.3 Design Fundamentals 602
21.3.1 Design Step 1: Identify the objects to be modeled 602
21.3.2 Design Step 2: List the properties and behaviors of each type of object 603
21.3.3 Design Step 3: Model properties with instance variables 605
21.3.4 Design Step 4: Model behaviors with methods 608
21.3.5 Design: The Process of Refinement 611
21.3.6 Filling in the Details 612
21.3.7 Summary of the Design Process 619
21.4 Incremental Testing and Debugging 620
21.4.1 Developing and Testing Individual Classes 620
21.4.2 Putting the Pieces Together: Testing the Controller 622
21.5 Classes That Don't Model the Real World 624
21.5.1 An Animated Shell Game 624
21.5.2 Identifying Classes That Don't Represent Entities in the Real World 625
21.5.3 Completing the Design for the Animated Shell Game 625
21.5.4 Filling in the Details for the Animated Shell Game 629
21.6 Guidelines for Writing Comments 632

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

21.7 Encapsulation and Information Hiding 635
21.7.1 Nested Rectangles: A Graphical Object Class 635
21.7.2 The Important Relationship between Encapsulation and Abstraction 637
21.8 Summary 638
21.9 Chapter Review Problems 640
21.10 Programming Problems 640

A Style Guidelines 643

A.1 Commenting 643
A.2 Blank Lines 644
A.3 Names 644
A.4 Format 645

B Objectdraw API Summary 646

B.1 `WindowController` 646
B.1.1 Methods to define in extensions of `Controller` or `WindowController` 646
B.1.2 Methods to define in extensions of `WindowController` 646
B.1.3 Methods to call in classes extending `Controller` or `WindowController` 647
B.2 `ActiveObject` 647
B.2.1 Methods to define in extensions of `ActiveObject` 647
B.2.2 Methods to call in classes extending `ActiveObject` 647
B.3 `Drawable` Objects 647
B.3.1 Constructors for `Drawable` objects 647
B.3.2 Methods for all `Drawable` objects 649
B.3.3 Methods for `Line`s only 649
B.3.4 Methods for rectangles, ovals, `VisibleImage`s, and `Text` objects 650
B.3.5 Methods for rectangles, ovals, and `VisibleImage`s only 650
B.3.6 Methods for `Text` objects only 650
B.4 Auxiliary Classes 650
B.4.1 Constructors for auxiliary objects 650
B.4.2 Methods for `Location` objects 651
B.4.3 Method for `RandomIntGenerator` objects 651
B.4.4 Method for `RandomDoubleGenerator` objects 651
B.5 Type and Interface Hierarchies 651
B.5.1 Subclass hierarchy for geometric objects 651
B.5.2 Subinterface hierarchy for geometric objects 652
B.5.3 Subclass hierarchy for controllers 652

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

C Navigating Java APIs 653

C.1 Introduction to objectdraw Documention 653
C.2 Getting Information about a Class 654
C.2.1 Summary Information about Constructors and Methods 655
C.2.2 Detailed Information about Constructors and Methods 657
C.3 When Should I Refer to an API? 658
C.4 Other Java APIs 658

D Programming without objectdraw 660

D.1 Extending `JApplet` rather than `Controller` 660
D.2 Extending `JApplet` rather than `WindowController` 662
D.3 Java Threads Can Replace `ActiveObject` 663
D.4 Java Graphics 664
D.5 Random Number Generators 667
D.6 Applications vs. Applets 668
D.6.1 Modifying an Applet to be an Application 668
D.6.2 Running an Applet from within a `JFrame` 669

Index 672

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

List of Figures

0.1 Chapter dependencies after core introductory topics xxxiv

1.1 Connect dots 1 through 82 5
1.2 Window displayed by a very simple program 8
1.3 Our first Java program 9
1.4 Changing the information displayed in the window 12
1.5 An Eclipse project window 14
1.6 A BlueJ project window 14
1.7 Entering the text of `TouchyWindow` under Eclipse 15
1.8 Entering the text of `TouchyWindow` under BlueJ 16
1.9 Running a program under BlueJ 16
1.10 Comparison of computer and Cartesian coordinate systems 17
1.11 Text enlarged to make pixels visible 18
1.12 A program window and its drawing coordinate axes 19
1.13 Drawing of a single line 21
1.14 A line from (100,150) to (200,0) 21
1.15 A program that draws two crossed lines 22
1.16 A `FilledOval` nested within a `FramedRect` 23
1.17 A simple program with instructions 26
1.18 Eclipse displaying a syntax error message 27
1.19 BlueJ displaying a syntax error message 28

2.1 An oval rises over the horizon 34
2.2 Declaring `sun` in the `RisingSun` program 36
2.3 Code for rising sun example 38
2.4 Rising sun program with reset feature 40
2.5 An application of the `translate` method 47

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

2.6 Display after one click 48
2.7 Display after second click 48
2.8 Display after many clicks 49
2.9 Tonight's forecast: Partly cloudy 50
2.10 Tonight's forecast: Partly moony? 50
2.11 Using a different parameter name 52
2.12 A program to record mouse-button changes 53
2.13 Connecting the ends of a mouse motion 54
2.14 A program to track mouse actions 55
2.15 Scribbling with a computer's mouse 56
2.16 A simple sketching program 58

3.1 Program to make the sun scroll with the mouse 64
3.2 Drawing produced by the `DrawGrid` program 65
3.3 The sun rises over the horizon 69
3.4 Program to make the sun scroll with the mouse 70
3.5 Drawing desired for Exercise 3.4.1 71
3.6 Using a numeric instance variable 74
3.7 A computer counting program takes its first step 79
3.8 A simple counting program 81
3.9 Counting in the Java console 82
3.10 Counting using the Java console 82
3.11 Sample message drawn by dice simulation program 86
3.12 Simulating the rolling of a pair of dice 87
3.13 Display for `ICanCountALot` 90
3.14 Initial display for `GrowMan` 91
3.15 Display for `GrowMan` after ten clicks 91

4.1 Screen shot of `Voting` program 93
4.2 Code for `Voting` class 94
4.3 Semantics of the `if-else` statement 96
4.4 Code to display individual and total vote counts 96
4.5 Semantics of the `if` with no `else` 97
4.6 Code for dragging a box 100
4.7 Three stages of dragging a rectangle 101
4.8 Voting for four candidates 106
4.9 A summary of the `boolean` and comparison operators in Java 108
4.10 `Craps` class illustrating nested conditionals 113
4.11 Display for `InvisibleBox` 118
4.12 Display for `Dicey` 119

5.1 Pattern constructed by `DrawGrid` 122
5.2 Pattern drawn by `DrawGrid` program after modifications 123
5.3 Drawing a grid without `Locations` 124
5.4 A program to test for steady clickers 127
5.5 Java program to measure mouse click duration 138

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

5.6 Sample output of the ClickTimer program 139
5.7 A program to display Morse code as dots and dashes 144
5.8 Sample of Morse code program display 145
5.9 A right-angle triangle 148
5.10 Display for DNAGenerator 149

6.1 Funny face generated by FaceDrag 152
6.2 Code for dragging a funny face 153
6.3 Revised code for dragging a funny face 154
6.4 FunnyFace class 162
6.5 Timer class and diagram 168
6.6 Chase class: instance variables and begin method 169
6.7 Chase class: onMouseClick method 170
6.8 FunnyFace constructor with a smile 171
6.9 A revised version of the smiling FunnyFace constructor using local variables 172
6.10 Chase class: revised onMouseClick method using local variable 173
6.11 Writing overloaded methods in terms of one version 174
6.12 A small stick-man 178
6.13 FunnyFace with a TopHat 179

7.1 Grass 181
7.2 Code for drawing the grass scene 182
7.3 Alternate code for drawing grass. Note the pattern that has emerged in the code 183
7.4 Code that allows a user to draw the grass in the grass scene 184
7.5 Using a while loop to draw blades of grass 185
7.6 Bricks 186
7.7 A while loop to draw a row of bricks 186
7.8 Grid 187
7.9 A while loop to draw the grid 188
7.10 The picture for the AlmostBullseye exercise 188
7.11 A while loop to draw a row of exactly ten bricks 189
7.12 Pictures of rulers for Exercises 7.3.1 and 7.3.2 190
7.13 A simple brick wall 191
7.14 Nested loops to draw a brick wall 192
7.15 Nested loops to draw a better brick wall 193
7.16 A better brick wall 194
7.17 Choosing an activity on a summer afternoon 200
7.18 Nested conditionals to choose an activity on a summer afternoon 201
7.19 Using a switch statement to randomly select a color 206
7.20 Generating an error message for unexpected cases 208
7.21 Example of scarf for Exercise 7.11.1 210
7.22 Bullseye for Exercise 7.11.3 211
7.23 Fence for Exercise 7.11.4 212
7.24 Railroad tracks for Exercise 7.11.5 212

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

8.1 FunnyFace and RevFaceDrag classes 215
8.2 Chase class: onMouseClick method 216
8.3 Chase class: revised onMouseClick method using private method resetGame 217
8.4 Scope example 221
8.5 Example using a static variable 223

9.1 A falling ball in motion 227
9.2 Class that creates a falling ball with each mouse click 228
9.3 Code for a FallingBall class 229
9.4 A drawing of a raindrop 230
9.5 Class that creates a falling raindrop picture with each mouse click 231
9.6 Code for a FallingRainDrop class 232
9.7 Defining a class that allows a user to drop a ball that changes color 233
9.8 Adding stopping functionality to a falling ball 235
9.9 A pool filling with rain 236
9.10 Adding a collector to collect raindrops 237
9.11 Making a droplet fill a collector 238
9.12 A rain cloud class 239
9.13 A class to generate movie credits 241
9.14 A class of individual movie credit lines 242
9.15 Making movement smoother 243
9.16 Guaranteeing minimum spacing between moving objects 245
9.17 Run of HitTheTarget 247

10.1 FunnyFace class 251
10.2 StraightFace class 252
10.3 TwoFaceDrag class 253
10.4 ShapeGenerator class, part 1 259
10.5 ShapeGenerator class, part 2 260
10.6 ShapeController class controls user interaction 261

11.1 JTextField in window 268
11.2 TextController class using JTextField 269
11.3 Layout of a window using BorderLayout layout manager 271
11.4 TextButtonController class using JTextField 273
11.5 FallingBall class 277
11.6 Menu constructed by MenuBallController 278
11.7 MenuBallController class 279
11.8 Using a panel to group GUI components 283
11.9 ButtonsBallController with multiple buttons in a panel: Part 1 284
11.10 ButtonsBallController with multiple buttons in a panel: Part 2 285
11.11 A panel with GridLayout manager to group buttons 287
11.12 A window with a Slider to set the speed 288
11.13 SliderBallController class with slider 289
11.14 JSlider with JLabel in JPanel 291
11.15 LabelBallController class with label showing speed 293

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

11.16 JTextArea in window 294
11.17 JTextArea in window 296
11.18 Window with text field, text area, and button 297
11.19 Controlling ball speed with the keyboard 299

12.1 Nested rectangles 306
12.2 A view of nested rectangles as a recursive structure 307
12.3 Recursive version of NestedRects class 308
12.4 Nested rectangles drawn by evaluating new NestedRects(50, 50, 19, 21, canvas) 310
12.5 Recursive version of NestedRects classes with interface 313
12.6 Objects constructed by evaluating new NestedRects(54, 54, 11, 13, canvas) 314
12.7 UrlListInterface interface 315
12.8 EmptyUrlList class 316
12.9 NonemptyUrlList class 317
12.10 A broccoli plant 323
12.11 Broccoli interface 324
12.12 Flower class implementing BroccoliPart 325
12.13 Broccoli class 326
12.14 BroccoliDrag class 327
12.15 Towers of Hanoi puzzle with 8 disks 332
12.16 Parsley 336

13.1 Checkerboard program output 343
13.2 Craps class illustrating do-while loops 346

14.1 Constructor for the Triangle class 360
14.2 Elements of the edge array refer to components of a Triangle 361
14.3 A straightforward implementation of the hide method for Triangles 361
14.4 Using a loop to implement the hide method for Triangles 362
14.5 Methods to move and show Triangles 363
14.6 Definition of the getVertices method 363
14.7 Vernon Hills S.M.A.R.T. 366
14.8 The vehicleReport method 368
14.9 A graph displaying numbers of speeders detected at different times of the day 369
14.10 The speedersSeen method 370
14.11 A drawing of the essence of a histogram 370
14.12 Method that draws histograms 371
14.13 A bar graph with rather short bars 371
14.14 Method to find largest value in speedersAt array 372
14.15 Method that draws well-scaled histograms 373
14.16 Sample of output expected from the drawLineGraph method 374
14.17 List of finishing times for racers 374
14.18 Definition of RacerInfo class 377
14.19 State of racer array and racerCount after adding 10 racers 378

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

14.20 The addRacer method 379
14.21 The individualResults method 381
14.22 A version of the getPlacement method 382
14.23 Alternate version of getPlacement 383
14.24 The teamScore method computes the score for one team 384
14.25 The teamStandings method 386
14.26 racer array with a missing element 388
14.27 racer array with entries after the omission shifted right 388
14.28 racer array after insertion is complete 389
14.29 racer array after moving element 9 389
14.30 racer array after moving elements 8 and 9 390
14.31 racer array after moving six elements over by one 391
14.32 racer array after executing racer[5] = racer[4]; 392
14.33 Array after executing racer[6] = racer[5]; 393
14.34 Where did all the racers go? 393
14.35 The addRacerAtPosition method 394
14.36 Array after executing racer[6] = null 394
14.37 racer array with last element duplicated 395
14.38 The removeRacerAtPosition method 396
14.39 The removeRacerAtPosition method with the loop reversed 397
14.40 A class to represent continents 398
14.41 Method mystery for Exercise 14.9.2 399
14.42 Skeleton of a Polynomial class for Exercise 14.9.3 400
14.43 Matching patches program after one patch is selected 404
14.44 Matching patches program after one pair has been identified 404
14.45 Matching patches program after many pairs have been identified 405

15.1 Interface used to enter new calendar events 407
15.2 A 12-element array. Each element in the array has the potential to refer to an array of strings 408
15.3 Daily events organized as a 12-element array (months) of arrays (days of each month) 409
15.4 Yearly calendar after two events have been entered 411
15.5 Another view of daily events as a jagged table 412
15.6 General structure of nested for loops to traverse a two-dimensional array 413
15.7 The YearlyCalendar class 414
15.8 Calendar entries for one day 415
15.9 A partial CalendarDay class 416
15.10 Magnified pixels from an image 418
15.11 A chessboard and sliding block puzzle: examples of objects that can be represented by two-dimensional matrices 418
15.12 Magic squares 419
15.13 Wrapping around after falling off the bottom of the square 424
15.14 Moving up if the next cell to be filled is already full 424
15.15 A method to fill a magic square 425
15.16 Digitization of an image 426

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

15.17 Magnified region of an image to be digitized. If one value is to represent this region, information will be lost 426
15.18 A pixel and its immediate neighborhood to be considered by the averaging algorithm 427
15.19 An image before and after smoothing 429
15.20 A simple smoothing algorithm 430
15.21 Player moving off the right of the game grid re-enters from the left 432

16.1 A program to display Morse code as dots and dashes 438
16.2 Instructions for the Morse code program 438
16.3 Using `\n` to force line breaks 439
16.4 `URLHistory` class 440
16.5 The `contains` method 441
16.6 Case-insensitive `URLHistory` class 444
16.7 A method to find all possible completions for a partial URL 446
16.8 Case-insensitive `URLHistory` class with trimmed URLs 448
16.9 Adding a URL to a history of URLs maintained in alphabetical order 449
16.10 Inserting a string into a list in alphabetical order 449
16.11 Character array containing the sequence of letters in the title of this book 452
16.12 User interface for a medical record system 453
16.13 A method to check whether a string looks like a valid integer 454
16.14 A partial Morse code alphabet 455
16.15 A method to translate a message into Morse code 455
16.16 A method to translate a single character into Morse code 456
16.17 Additional Morse code symbols 457

17.1 The user interface of an interactive card game 464
17.2 An empty frame 465
17.3 Implementation of an empty framed display class 466
17.4 Definition of the `FramedText` subclass 468
17.5 A `FramedBarMeter` class that references a `protected` superclass member 473
17.6 `FramedDisplay` class with added `protected` accessors 474
17.7 Definition of the `FramedBarMeter` class 475
17.8 Revision of the `FramedText` subclass 476
17.9 Definition of `FramedCounter` as an extension of `FramedText` 478
17.10 Inheritance hierarchy for subclasses of `FramedDisplay` 478
17.11 Definition of `FramedText` with specialized `highlight` method 483
17.12 Version of `FramedDisplay` based on `positionContents` 490
17.13 `FramedText` with revised `positionContents` 491
17.14 Definition of the `FramedBarMeter` class 492
17.15 The `RubberBall` and `TennisBall` classes for Exercises 17.8.3 and 17.8.4 496
17.16 The `SimpleSticky` and `MessageSticky` classes for Exercises 17.8.7 and 17.8.8 499
17.17 Using the `SimpleButton` and `ButtonAndWindowController` classes 503

18.1 Interface used to enter new calendar events 507
18.2 Cautious code to add a calendar entry 508

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

18.3 Exception handling with a `try-catch` statement 508
18.4 `setEvent` method that takes `String` parameters 511
18.5 Event-handling code that passes `String`s to `setEvent` 511
18.6 Methods for Exercise 18.3.1 512
18.7 A `try-catch` with two `catch` clauses 513
18.8 Example of nested `try-catch` statements for Exercise 18.4.1 514
18.9 A portion of the `Exception` inheritance hierarchy 515
18.10 A `try-catch` that catches all `RuntimeException`s 516
18.11 Failing to handle a checked exception 518
18.12 Handling a checked exception 518
18.13 A method with a `throws` clause 519
18.14 `setEvent` revised to throw an `IllegalArgumentException` 520
18.15 `setEvent` with two `throw` statements 521
18.16 Definition of an `Exception` class 522
18.17 Interface for program described in Exercise 18.10.1 525

19.1 A class of objects designed to represent single bookmarks 529
19.2 Method to display all bookmarks on the screen 530
19.3 Skeleton of code to handle I/O exceptions associated with streams 532
19.4 Definition of the `saveBookmarksFile` method 532
19.5 Browser display of file produced by `saveBookmarksFile` 533
19.6 Browser display of an actual bookmarks.html file 533
19.7 Source view of HTML for web page shown in Figure 19.6 534
19.8 Definition of the `toBookmarkFileEntry` method 535
19.9 A completed version of the `saveBookmarksFile` method 537
19.10 Possible implementation of `println` for `String`s 538
19.11 The `retrieveBookmarksFile` method 541
19.12 `DisplayFile` program for Exercise 19.2.1 542
19.13 User interface of program for Exercise 19.2.1 543
19.14 A minimal application to create a window 544
19.15 Window displayed by `JustAWindow` program 545
19.16 Interface provided by `TimeClock` application 545
19.17 Definition of a `TimeClock` application 547
19.18 Interface for the program described in Exercise 19.3.1 548
19.19 Main window of the `SimpleEdit` program 549
19.20 Instance variables and `begin` method for `SimpleEdit` 550
19.21 The `loadText` method of `SimpleEdit` 551
19.22 The `openInput` method 551
19.23 Dialog displayed by `showOpenDialog` 552
19.24 Methods used by `SimpleEdit` to save files 553
19.25 Screen displayed by program for Exercise 19.4.1 553
19.26 Window displayed by the Simple Weather program 557
19.27 A temperature graph for Fairbanks from January, 2004 558
19.28 Website of the Alaska Climate Research Center 559
19.29 A method to display the HTML of the Alaska Climate Center web page 560
19.30 The `sendRequest` method 560

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

19.31 The `displayResponse` method 560
19.32 A portion of the HTML for `http://climate.gi.alaska.edu/` 562
19.33 The `getTemp` method 563
19.34 The `processResponse` and `getLineContaining` methods 563
19.35 Output format desired for Exercise 19.8.1 571
19.36 Sample output from program for Exercise 19.8.3 571
19.37 Addresses of Daytime Protocol servers 572

20.1 Iterative version of linear search of an `int` array 574
20.2 Recursive version of linear search 575
20.3 Recursive binary search 579
20.4 Iterative binary search 581
20.5 Iterative selection sort and helping methods 584
20.6 Recursive selection sort 587
20.7 Iterative insertion sort 589
20.8 Recursive insertion sort 591
20.9 Recursive merge sort 594
20.10 Merge method used in merge sort 595

21.1 User interface for a simple shell game 603
21.2 First level of design for a `Cup` class: properties and behaviors 604
21.3 First level of design for a `Marble` class: properties and behaviors 605
21.4 First level of design for a `ShellGame` class: properties and behaviors 605
21.5 Second level of design for a `Cup` class: modeling properties with instance variables 606
21.6 Second level of design for a `Marble` class: modeling properties with instance variables 607
21.7 Second level of design for a `ShellGame` class: modeling properties with instance variables 607
21.8 Third level of design for a `Cup` class: modeling behaviors with methods 609
21.9 Third level of design for a `Marble` class: modeling behaviors with methods 610
21.10 Third level of design for a `ShellGame` class: modeling behaviors with methods 611
21.11 Filling in the details for a `Cup` class: instance variables and constructor 613
21.12 Filling in the details for a `Cup` class: methods 614
21.13 Filling in the details for a `ShellGame` class: instance variables and `begin` method 616
21.14 Filling in the details for a `ShellGame` class: mouse-event-handling methods 617
21.15 A simple controller to test that a cup can be dragged 622
21.16 A simple controller to test that a cup with a marble can be dragged and its contents revealed 623
21.17 Extra methods required by the `Cup` class 626
21.18 High-level design for a `Shuffler` class 626
21.19 Animation of shuffling 628
21.20 Filling in the design details for the `Shuffler` class 629
21.21 Design of the controller class for the animated shell game 630
21.22 Filling in the details of the `Shuffler` class: instance variables and constructor 631
21.23 Implementation of shuffling 633

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

21.24 A simple program to draw and move nested rectangles 637
21.25 A sample drawing produced by running RectangleDrawer 638
21.26 An array-based `NestedRects` class 639
21.27 Leaky faucet and glass before the faucet drips 641
21.28 A Hangman game 642

C.1 Viewing the objectdraw API in a web browser 654
C.2 Clicking on the `FilledRect` class gives information on `FilledRect`s 655
C.3 A summary of constructors 656
C.4 Detail of a constructor 657
C.5 Method summary for the `FilledRect` class 657
C.6 Detail of a method 658
C.7 Detail of all constructors for `FilledRect` 659

D.1 Handling mouse events in an extension of `JApplet` 663
D.2 Extending a `Thread` instead of `ActiveObject` 664
D.3 Example using standard Java graphics 666
D.4 Application with mouse event handling 669
D.5 Class that can contain an applet 670

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

Preface

A Java-based introductory course provides new challenges to instructors and students. While Java is simpler than C++, the fact that Java is an object-oriented language with a significant number of standard libraries adds both new complexities and opportunities.

This introductory computer science text provides a new approach to teaching programming in Java that combines several interesting features:

1. an objects-first approach to programming,
2. the intensive use of object-oriented graphics,
3. the use of concurrency early,
4. the use of event-driven programming from the beginning.

At first glance, this list of topics might seem overwhelming for an introductory text, but the synergy of these features results in a surprisingly effective introduction to programming using Java, especially when presented with the help of a library, objectdraw, that we have developed.

0.1 Target Audience

The primary target audiences for this text are first-year computer science majors, other college and university students interested in programming, and students taking high school advanced placement courses in computer science. In particular, this text covers the AP A exam material.

0.2 For the Student

0.2.1 Mysterious Buzzwords

We began this preface by listing some of the special features of this text that we find particularly exciting. Specifically, we said that the text provides:

1. an objects-first approach to programming,
2. the intensive use of object-oriented graphics,

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

3. the use of concurrency early,
4. the use of event-driven programming from the beginning.

At this point you're probably wondering what all of these buzzwords mean.

Java is an example of an *object-oriented* programming language. Just as there are many different types of spoken languages, there are many different computer programming languages. The object-oriented languages are simply one class of languages. Because programming languages differ from each other in many ways, it stands to reason that they should not all be taught in the same way. Since Java is object oriented, we have aimed to present it in a manner that is appropriate for that language paradigm.

From the beginning of the text, you will learn how to write programs that involve simple graphics—rectangles, ovals, and lines, for example. You will even learn to write programs that create graphical animations, using mechanisms for *concurrency*. We find that both students and instructors enjoy writing programs that involve interesting, albeit simple, graphics. In addition to being fun, graphics are very concrete. When a program involves drawing and manipulating graphical objects in a window, you can actually see what the program is doing. We find this helpful for the beginning programmer.

We also introduce *event-driven programming* early. While you probably haven't heard the term "event-driven programming", you're almost certainly familiar with it. If you've interacted with a computer by pulling down menus, clicking on icons, and dragging items with a mouse, you've interacted with event-driven programs. The programs you will learn to write will allow a user to interact with them through mouse movements, buttons, scroll bars, and so on.

0.2.2 How to Read This Book

Practice is an extremely important component of learning to program. Therefore, we have provided many opportunities for you to practice as you read this text. Each chapter contains embedded exercises that will allow you to check your understanding. Read with a pencil and paper beside you, so that you can do these short exercises as you go along. In addition, at the end of each chapter you will find chapter review exercises as well as programming problems. Working through the review exercises will help you determine whether you have understood the major concepts introduced in the chapter. Once you feel comfortable with these, try the programming problems. The more you do, the better you'll know Java.

0.3 For the Instructor

0.3.1 Special Features of This Text

A Graphics Library for the Objects-First Approach

We have adopted the use of graphics for our first examples and have constructed a truly object-oriented library of graphics classes. There are several reasons we believe that graphics provide a good setting for introducing object-oriented programming.

First, graphics are good examplars of objects. Graphics classes (e.g., framed and filled rectangles and ovals) provide good examples of objects because they have state (their location and dimensions) and a useful collection of methods that go well beyond methods that just get and set instance variables. Second, the graphics classes in our objectdraw library provide excellent

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

visual feedback for novice programmers. When a graphics object is created, it appears on the screen immediately. When a graphics object in our library is moved or resized, the picture on the screen changes immediately. As a result, if a program contains a logical error, that error is immediately visible on the screen. Third, graphics provide motivating examples. With graphics, very simple programs can become much more interesting to students. Moreover, once animations are introduced, it is easy to provide fun and interesting examples well before the introduction of arrays. Finally, graphics persist in the course. Rather than introducing a set of example objects and then discarding it, an instructor can use the graphics library throughout the course.

Our objectdraw library not only provides the graphics classes, it also provides a `WindowController` class that extends `JApplet` by installing a `DrawingCanvas` in the center of the window. `DrawingCanvas` is an extension of `JComponent` that keeps track of the objects on the canvas and redraws them whenever necessary. This reduces the complexity of using graphics for novice programmers.

Event-Driven Programming

Some authors have argued for an event-driven approach in an introductory course because “real” programs that students use every day operate in an event-driven way. In students’ use of computers they rarely see programs that respond to line-by-line text input. Thus event-driven programming is more motivating for students.

We believe there are several other pedagogically important advantages to an event-driven approach in an introductory course. One very important advantage is that students get experience writing methods from the beginning. Moreover, the methods tend to be very short.

In our library, we provide an environment in which novices learn to program by defining simple mouse-event-handling methods. For example, our `onMouseDrag` method is similar to standard Java’s `mouseDragged` method, except that it has a simpler parameter. Because it is called repeatedly while the mouse is being dragged, very interesting programs can be constructed without using loops. This use of event-driven programming allows us to postpone the discussion of loops until after we discuss the definition of classes, while still presenting interesting examples to students.

Students get experience writing methods and using parameters by writing methods with fixed names and numbers of parameters, simplifying the introduction of these concepts. For example, all of our mouse-handling methods take a single `Location` parameter representing where the mouse is when the event occurs. Students become accustomed to using these formal parameters inside the associated method bodies. At the same time, students use actual parameters in the graphics commands.

This experience in writing event-handling methods with well-specified names and signatures, as well as the experience of writing code to send messages with actual parameters to graphic objects, makes the transition to designing and writing classes and their methods easier for students. Students still need to work to understand the “how" and “why" of parameter passing, but they will have seen and written many examples. That helps students in writing and understanding their own classes.

Objects-First

The combination of graphics and event-driven programming supports our objects-first approach. Students see example programs using objects from the graphics library starting from the first

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

chapter of the text. Examples contain code to create new graphics objects and send messages to them. Moreover, the programs are extensions of the `WindowController` class.

Because the `WindowController` class of our library is an extension of Java's `JApplet` class, students need not begin with the static `main` method, and then have to learn about the differences between static and nonstatic methods. Instead they write instance methods that respond to mouse events. Thus students are introduced to using objects and writing their own methods from the first chapter of the text.

In the sixth chapter, students learn how to write their own classes. This chapter occurs before the introduction of loops, and just after the introduction of conditional statements. Our approach using event-driven programming allows us to construct and use interesting classes at this early point in the text.

Concurrency Early

We found that when examples are properly chosen to avoid race conditions, concurrent programming is conceptually easy for students to understand. After all, the world is concurrent, so there is nothing unnatural to students in having several threads executing concurrently. Moreover, many applications are much easier to program using concurrency rather than as a single thread.

We have provided a class `ActiveObject` in our library that supports using and managing threads. From a student's point of view, the primary difference between the `ActiveObject` class that we provide and the built-in `Thread` class is that we provide a variant of the `sleep` method that does not throw exceptions. As a result we are able to introduce concurrency in Chapter 9 of the text, before our discussion of exceptions. Behind the scenes, we also manage threads so that when a program (or applet in a web page) terminates, all threads will be terminated gracefully.

0.3.2 Why Introduce a Library?

We have chosen to introduce a library to support our approach, because it reduces syntactic and conceptual complexity early in the text. While we depend on the library early, it is not our intention to teach a different style of programming than that normally supported by Java. Our philosophy is to provide support early, but to also teach students the "right" way to program in Java.

A possible obstacle to using event-driven programming early in Java is the number of language and library features that must be introduced in order to handle events. For example, one would have to introduce listeners, interfaces, Java events, and so on. Moreover, if a class is to implement, for example, `MouseListener`, then it must implement all of the mouse-listening methods, even if only one is needed in the program.

Our library reduces this complexity as the `WindowController` class from the library implements both of the mouse listener interfaces. It also provides event-handling methods that take the `Location` of the mouse, rather than a more general `MouseEvent`. The advantage of getting a `Location` (a library type representing a point on the screen, but using doubles rather than integers) is that the useful information is immediately available, rather than requiring the programmer to extract it first. Finally, with our library, students only need to write the event-handling methods that they actually use in their program.

In Chapter 11 we teach students about standard Java GUI components. In conjunction with this we also teach students the standard Java event model. Students learn to associate listeners with user interface components and to write methods to handle the generated events. Thus they

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

do learn how to program without using our library, but at a time when they are better equipped to understand the needed concepts.

As we noted above, introducing threads without using our library would require a discussion of exceptions before being able to pause a thread. Moreover, the exception that must be handled with the `sleep` method is a very bad first example of exceptions, as there is generally not much to do to handle it. Because we think exceptions can be better motivated later in the course (for example, in discussing I/O) and because they involve the complexity of inheritance and subtyping, we designed our library to enable us to postpone the discussion of exceptions. We do not use the library as an excuse to avoid teaching key components of Java. Instead we use it to provide a more pedagogically sound approach to presenting the various concepts introduced.

0.3.3 Supplementary Materials for Instructors

Supplementary materials are available on-line for instructors at `http://eventfuljava.cs.williams.edu`. These materials include the objectdraw library, a rich collection of sample programs and laboratory assignments that use the library and that are coordinated with the text, and detailed lecture notes.

The sample programs include those already in the text as well as a large collection of additional examples. The supplementary examples are rich and varied and add a great deal to the overall presentation of the material in the text. In some cases the additional programs stress (and therefore reinforce) certain dependencies. (This is in direct contrast to the way in which we wrote the text, where we attempted to minimize dependencies wherever possible.) Many of the additional examples involve `ActiveObject`s and, more specifically, animation. These do not always serve as the best types of examples in a text, as a book is a static medium, but they can be used extremely effectively by the instructor in a classroom or laboratory setting.

0.4 Flexibility for the Instructor and Student

In this text we have aimed to provide maximum flexibility for the reader. We expect the reader to cover the core introductory material in Chapters 1, 2, 3, 4, 6, 7, and 9. These chapters introduce the objectdraw library, conditionals, classes, `while` loops, and concurrency. Chapters 5 and 8 provide additional details about the topics introduced in Chapters 3 and 6. As well, they introduce strings and the topics of declaration and scope. While they are important, these chapters can be covered later, if desired.

Optional sections in all of the chapters, marked with an asterisk (*), can also be skipped.

The remainder of the text presents topics in the order in which we cover them in our course. We have found this order to work very well. However, as remarked earlier one of our goals was to avoid topic dependencies as much as possible, so that instructors could tailor their courses as appropriate for their students and their institutions. Figure 0.1 shows the ways in which Chapters 10–19 depend upon each other. Note that, in particular, recursion (and recursive data structures) and arrays can be presented in any order. It is also possible to cover inheritance before either recursion or arrays.

In addition to the dependencies shown in Figure 0.1, it is important to note that later chapters assume knowledge of GUI to a limited extent. In particular, examples in these later chapters make use of `JTextField`s and the `setText` and `getText` methods.

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

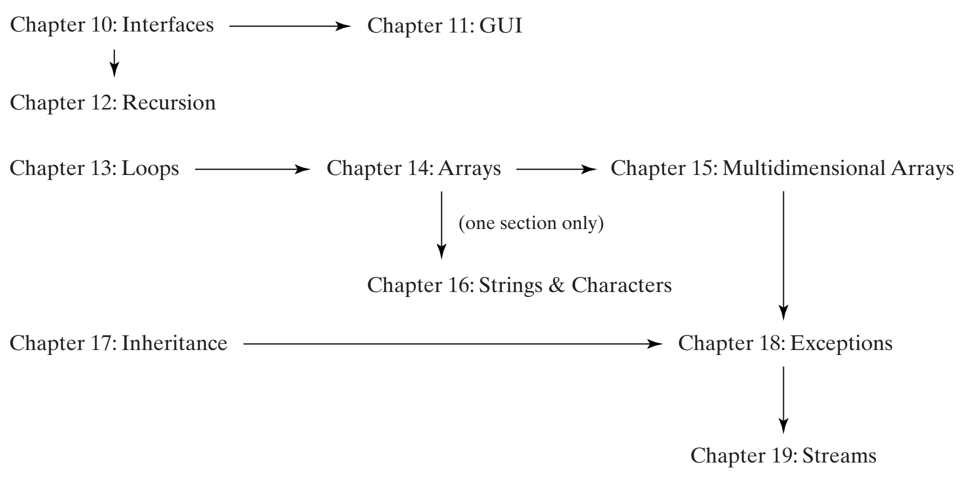

Figure 0.1 Chapter dependencies after core introductory topics

Chapter 20. "Searching and Sorting", is an advanced topic and assumes knowledge of arrays and recursion. The sections on search do, however, give both iterative and recursive versions of the algorithms presented.

The second half of Chapter 21, "Introduction to Object-Oriented Design", assumes that students are familiar with both recursion and arrays. However, the first half of that chapter assumes nothing beyond familiarity with the core chapters, 1–9. If desired, the reader could work through the design unit in two stages, covering the first half after the core introductory material and the second half after the more advanced topics.

Finally, it is important to note that this text is not meant to be a complete reference on the Java programming language. We have strived to present the elements of the language at a level that is appropriate for a beginner. Some of the chapters in the second half of the text, for example, provide introductions to important concepts, without necessarily providing details on the level that an advanced student might require. Our goal is to give students a firm footing, with the expectation that they will develop a deeper and more complete understanding of the language later, as they gain more experience.

0.5 Additional Practical Information

We have included a great deal of additional useful material in several appendices to the book. The first appendix provides style guidelines for programming. While there are fairly standard conventions followed by Java programmers, some issues of style are obviously subjective. Students should note that their instructors might provide their own guidelines.

The next appendix provides a summary of the classes and methods available in the objectdraw library. In the third appendix, we take the reader through the process of navigating the documentation for Java APIs. An API (Application Programming Interface) specifies the details a programmer must know to use the resources a library provides. We go through parts of the API

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.

for the objectdraw library, but as the documentation for many APIs follows a standard format, the reader should then be able to read other API documentation as well.

The transition from our library to standard Java is quite straightforward. The final appendix summarizes for the reader the standard Java equivalents for many of the features in the library.

0.6 Acknowledgements

This book would not have been possible without help and feedback from many people. First we thank our Williams College faculty colleagues, Barbara Lerner, Jim Teresco, Steven Freund, and Stacia Wyman, who were willing to teach with the sometimes very sketchy notes and not always stable library that we provided. Hundreds of Williams students enthusiastically met the challenge of learning a new way of approaching computer science using our materials as they developed over the last few years. Special thanks go to our undergraduate teaching assistants, many of whom worked hard to learn new material and helped make it seem easy to the students. A number of other Williams students worked with us in developing the library, writing up lab and homework problems and their solutions, testing the library, and designing the web pages for our course materials. These students include Peter Applegate, Jing Cao, Brendan Dougherty, Marcus Duyzend, Cheng Hu, Jonathan Kallay, Krishna Kannan, Christine Osterman, and Ashok Pillai.

A very important part of developing our materials was getting feedback from faculty at other institutions who used our materials in their own classes. While there are too many testers to list them all here, we single out the following for their very helpful feedback on the text and materials: Barbara Adler, Mary Courtney, Chris Haynes, David Housman, Lonnie Fairchild, James Taylor, and Douglas Troy. Special thanks to Chris Nevison for using the objectdraw library in his short courses for high school Advanced Placement Computer Science teachers.

The outside reviewers engaged by Prentice Hall provided very thoughtful comments on this text. While we did not follow all of their advice, their detailed feedback resulted in many improvements and clarifications in the text. The reviewers include: Chris Haynes of Indiana University, Richard Albright of Goldey-Beacom College, Henry A. Etlinger of Rochester Institute of Technology, Carlos A. Varela of Rensselaer Polytechnic Institute, Richard E. Pattis of Carnegie Mellon University, Ron Zacharski of New Mexico State University, Trudy Howles of Rochester Institute of Technology, Mark Williams of Lane Community College, Mary Courtney of Pace University, Mary Ann May-Pumphrey of De Anza College, David Housman of Goshen College, Lawrence C. Petersen of Texas A & M University, Mark A. Holliday of Western Carolina University, Gavin T. Osborne of Saskatchewan Institute of Applied Science and Technology, and Adel S. Elmaghraby of University of Louisville.

We are especially grateful to our editors from Prentice Hall, Petra Recter, who signed us up, Kate Hargett, who took over after Petra moved to a new division, and who gave birth to her first child about the time we sent this manuscript off into the publisher's arms, and Tracy Dunkelberger, who saw this project through to completion. We are also grateful to our production editor, Irwin Zucker.

We hope you enjoy using this book as much as we have enjoyed developing this approach to an introductory course. We appreciate receiving comments and suggestions on this text and the associated materials. Contact information is available on our web page:

`http://eventfuljava.cs.williams.edu`

Frontmatter from JAVA: AN EVENTFUL APPROACH by Bruce, Danyluk, and Murtagh, © 2006 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved. This material is protected under all copyright laws as they currently exist. No portion of this material may be reproduced, in any form or by any means, without permission in writing from the publisher.