To Christina
FEATURES OF THIS TEXT

FAQ: Why Do We Need new?
When new is used in an expression such as the following, you can think of it as creating the instance variables of the object.

```java
SpeciesFirstTry speciesOfLastMonth = new SpeciesFirstTry();
```

An object of a class type, such as speciesOfLastMonth, can have smaller variables inside of it, namely, the instance variables of the object. The new places these instance variables inside of the object. We will explain this use of new more completely in Section 4.3.

Gotcha

Helps students identify possible programming pitfalls.

Quick Reference: Method Invocation (Calling a Method)
You invoke a method by writing the calling object followed by a dot, then the name of the method, and finally a set of parentheses that may (or may not) have information to pass to the method.

Java Tip

Gives students helpful information about the Java programming language.

Quick Reference

Provides concise, user-friendly reference for key programming concepts.

Java Tip

Use of return in void Methods

A `void` method returns no value and so is not required to have any return statement. However, there is a kind of return statement that you may sometimes want to use in a `void` method. A return statement within a `void` method has the form

```java
return;
```

Remember

Reinforces concepts presented in the chapter.

Self-Test Questions

Provides students the opportunity to practice skills learned in the chapter.

1. Consider the program in Display 4.4. Suppose you wanted to add another species object called speciesOfTheYear, and suppose you wanted the user to give it data, specifically a name, population, and growth rate. What code do you need to add to the program? (Hint: It requires only three or four lines of code.)

2. Suppose `Employee` is a class with a `void` method named `readInput` and `dilbert` is an object of the class `Employee`. So `dilbert` was named and created by the following:

```java
Employee dilbert = new Employee();
```
Preface for Instructors

This book was designed to be used in a first course in programming and computer science. It covers programming techniques, as well as the basics of the Java programming language. It is suitable for courses as short as one quarter or as long as one full academic year. No previous programming experience is required, nor is any mathematics, other than a little high school algebra. The book can also be used for a course designed to teach Java to students who have already had another programming course, in which case the first few chapters can be assigned as outside reading. (For students who have had previous programming experience in C or C++, there is an appendix that explains some of the differences between Java and C or C++.)

All of the code in the book has been tested with Java 2 from Sun Microsystems (version 1.4). To be fully compatible with the material presented, the Java used in your class should be version 1.4 or higher. The coverage of Java was carefully arrived at by class testing and is a concise, accessible introduction for beginners.

Changes in This Edition

If you have not used the second edition of this text, you can skip this subsection. If you have used the second edition, this subsection will tell you how this third edition differs from that edition.

For instructors, the transition from the second edition of the text to this third edition is easy: You can teach the same course, with basically the same topics presented, in the same order and with only very minor changes in the material covered.

In addition to technical changes in the material, which we discuss shortly, we have completely gone over and refined the presentation for this edition. Without abandoning our student-friendly, easy-to-understand style, we have redone the entire text to make it more concise. Some have complained about the quality of the copyediting in the previous edition. We have gone to great pains to obtain a copy editor we have used before and know to be an excellent editor. Here are the main differences between this and the previous edition:

We have added coverage of the new assertion-checking facility now built into Java. This topic is covered in Chapter 4.

Binary file I/O is now covered using the classes `ObjectInputStream` and `ObjectOutputStream`, rather than the classes `DataInputStream` and `DataOutputStream`. For I/O of primitive types and strings, the details are the same in either case. For I/O of class objects, `ObjectInputStream` and `ObjectOutputStream` are more object oriented and, even more important, are easier to use and perform better.

The previous edition allowed for covering either text files first or binary files first. This required some repetition of material. Many instructors disliked the repetition, and as far as we can now tell, nobody was choosing the option of covering binary files first. This edition requires that text files be covered before binary files. As a result, the discussion is more concise and is organized in the way that instructors prefer. (You can, of course, skip coverage of binary files completely if you wish.)
This edition adds coverage of the Unified Modeling Language (UML). All UML coverage is optional (even though it is not labeled as such).

The material on JOptionPane that was in Chapter 2 of the second edition has been moved to Appendix 10 in this edition. Instructors told us that they did not cover the material, so we removed it from Chapter 2 to simplify that chapter. For any instructors who do want to teach the material, we left it in an appendix that can be covered anytime after Chapter 2.

Chapter 15, which covered more advanced graphics techniques, such as drawing pictures, advanced color manipulation, and control of fonts, has been eliminated. Instructors told us that they never used the material and that they would prefer a shorter book. Chapters 12, 13, and 14 still include extensive coverage of Swing windowing interfaces and applets.

The CD in the book now comes with Java 2 SDK edition, version 1.4, for Windows and Linux and with Sun ONE Studio 4, Community Edition Integrated Development Environment IDE (formerly Forte for Java 4, Community Edition), for Windows and Linux. The CD also contains a simpler shareware IDE named TextPad, which runs under Windows and which beginners may find more friendly and easier to learn. Because we have added Sun ONE Studio 4 (aka Forte), there was no longer any need for JBuilder, so we have omitted the trial version of JBuilder from the CD for this edition. JBuilder is a fine IDE and will work well with the text, but we felt that a third IDE choice might be too confusing for beginning students. If you prefer JBuilder, you can have your students download a trial version from Borland’s Web site (http://www.borland.com/ as we went to press).

Latest Java Coverage

This edition has been updated to use the latest features of Java, including assertion checking and the updated versions of the Swing and file I/O classes. As we wrote the book, we checked all code on Sun’s Java 2 version 1.4.

Flexible Sequence of Topics

If you are an instructor, this book adapts the material presented to the way you teach, rather than making you adapt to the material. The book does not tightly prescribe the order in which your course must cover topics, nor does it prescribe the specialized libraries that must be used in your course. You can easily change the order in which you cover chapters and sections. The details about rearranging material are explained in a dependency chart following the acknowledgements section. More details are given in a “Prerequisites” section at the start of each chapter.

Since Java does not include any simple console input, most texts (even those which are more advanced) provide a class for console input. This book adds only one simple class for console input, thus requiring as little nonstandard software as possible. Even that one class, which is included early in the book, becomes an understandable programming example for students well before the end of the book. All of the remaining software is from standard Java libraries that should be part of any Java installation.
Coverage of Problem-Solving and Programming Techniques

The book is designed to teach students basic problem-solving and programming techniques and is not simply a book about Java syntax. Numerous case studies and programming tips, as well as many other sections, explain important problem-solving and programming techniques, such as loop design techniques, debugging techniques, style techniques, abstract data types, and basic object-oriented programming techniques, including UML and event-driven programming, as well as other computer science topics.

Object-Oriented and Traditional Techniques

Any course that really teaches Java must teach classes early, since everything in Java involves classes. A Java program is a class. The data type for strings of characters is a class. Even the behavior of the equals operator (==) depends on whether it is comparing objects from classes or simpler data items. Classes cannot be avoided, except by means of absurdly long and complicated “magic formulas.” Accordingly, the book introduces classes fairly early. Some exposure to using classes is given in Chapters 1 and 2. Chapter 4 covers how to define classes. All of the basic information about classes—including inheritance—is presented by the end of Chapter 7 (even if you omit Chapter 6). However, some topics regarding classes (including inheritance) can be postponed to later in a course.

Although the book introduces classes early, it does not neglect traditional programming techniques, such as top-down design and loop design techniques. These older topics may no longer be glamorous, but they are information that all beginning students need.

UML Coverage

New to this edition is the coverage of UML, which starts in Chapter 4. Instructors who would prefer to leave UML coverage to a later course can skip the material without losing continuity of the text.

Swing GUIs and Applets

Starting with the first version of Java 2, Java comes with an improved graphical user interface (GUI) library known as Swing that allows programmers to design portable GUIs. This book uses Swing to teach students to produce professional-looking windowing interfaces. In the process, students learn event-driven programming, as well as receiving a lot of practice with object-oriented programming.

As we class-tested this material and gathered the views of instructors, we found that regular Swing GUIs were a more accessible way than applets to teach students object-oriented programming. Thus, we place greater emphasis on regular Swing GUIs in this text. Such an emphasis makes sense, since almost all advanced applet tools are really general Swing tools. However, for those who want to cover applets early, Chapter 1 has an optional section that previews them. Chapter 13 covers applets in detail and may be presented much earlier than the chapter number suggests. You may choose to introduce GUIs early, late, or not at all.
In addition to the optional GUI material in Chapter 1, there are three full chapters on GUIs, thus giving thorough coverage of Swing GUIs and applets.

**Language Details and Sample Code**

The book teaches programming technique, rather than simply the Java language. However, neither students nor instructors would be satisfied with an introductory programming course that did not also teach the programming language. Until you calm a student’s fears about language details, it is often impossible to focus her or his attention on bigger issues. For this reason, the book gives complete explanations of Java language features and lots of sample code. Programs are presented in their entirety, along with sample input and output. In many cases, there are even extra complete examples on the CD, in addition to the complete examples in the text.

**Self-Test Questions**

Self-test questions are spread throughout each chapter. These questions have a wide range of difficulty levels. Some require only a one-word answer, whereas others require the reader to write an entire nontrivial program. Complete answers to all the self-test questions, including those requiring full programs, are given at the end of each chapter.

**Class Tested**

The material in the book has been fully class tested. Much of the material and many of the methods of presentation were revised in response to this testing.

**Support Material**

The support materials described here that are not included with the book can be obtained from the publisher or over the Internet.

**Companion CD**

Each book contains a CD that includes all the programs and classes in the volume. The CD also includes Sun Microsystems’ Java 2 SDK edition, version 1.4, for Windows and Linux and Sun ONE Studio 4, Community Edition (formerly Forte for Java 4, Community Edition), for Windows and Linux. The CD also includes a copy of TextPad, a simpler shareware IDE that runs under Windows and that may be easier for students to deal with than the Sun ONE Studio 4 IDE (previously known as Forte).

**Instructor’s Resource Guide and Companion Web Site**

The instructor’s tools include a chapter-by-chapter *Instructor’s Resource Guide* that contains numerous teaching hints, quiz questions with solutions, and solutions to many programming exercises. The companion Web site includes code, PowerPoint slides, and other...
teaching resources. Instructors should contact their Prentice Hall sales representative to obtain a copy of the *Instructor’s Resource Guide* and to receive information on how to access the companion Website. For the name and number of your sales representative, call Prentice Hall Faculty Services at 1-800-526-0485. Additional information on this book and other Prentice Hall products can be found on Prentice Hall’s Web site at

http://www.prenhall.com/

**Other Resources**

There are many Java resources available from other vendors which we do not supply, but which you may wish to purchase. One such resource is the JJ environment from Public Static Void Main. This environment includes a simple IDE as well as course administration software all accessed via the Internet. The JJ environment has been setup to be highly compatible with this book. In particular, the *SavitchIn* class, which is used for keyboard input in this book, is an installed library class in JJ. For more information see their website:

http://www.LearnJavaNow.org/

Walter Savitch
wsavitch@ucsd.edu
http://www.cse.ucsd.edu/users/savitch
**Preface for Students**

This book is designed to teach you the Java programming language and, even more importantly, to teach you basic programming techniques. It requires no previous programming experience and no mathematics other than some simple high school algebra. However, to get the full benefit of the book, you should have a version of Java available on your computer, so that you can practice with the examples and techniques given. You should have a version of Java called Java 2. That “2” is not a version number, but something more like a generation number. The version number will be of the form 1.n.x, such as 1.2.x or 1.4.x. The “x” is another number and may not even be there. To be fully compatible with the book, your Java version number should be 1.4.x or higher. (The exact number that is filled in for the “x” is not critical. Indeed, the “x” need not even be present. If software says only “version 1.4,” that is fine.)

**If You Have Programmed Before**

You need not have any previous programming experience to use this book. It was designed for beginners. Nonetheless, you can still use the book to learn Java if you happen to have had experience with some other programming language—but allow us to give you a few words of advice. Do not assume that Java is the same as the programming language(s) you are accustomed to using. All languages are different, and the differences, even if small, are large enough to give you problems. Read at least the boxed material in Section 1.3 of Chapter 1 and all of the boxed material in Chapters 2 and 3. These boxed sections are labeled “Quick Reference,” “Remember,” and “FAQ.” By the time you reach Chapter 4, you would be wise to read the entire chapter.

If you have programmed before in either C or C++, the transition to Java can be troublesome. At first glance, Java may seem almost the same as C or C++. However, Java is very different from these languages, and you need to be aware of the differences. Appendix 11 presents a comparison of Java and C++ that will help you see what the differences are.

**Copies of the Programs in the Text**

Accompanying the book is a CD that includes all the programs and other software examples given in the volume, so that you can practice with these examples without having to type them into your computer.

**Obtaining a Copy of Java**

The version of Java you use depends somewhat on what operating system you are using. Be sure to consult the subsection which follows that corresponds to your operating system.

**Microsoft Windows**

The CD that comes with this text contains a copy of the Java 2 SDK edition, version 1.4, and the Sun ONE Studio 4 (also known as Forte). The Java 2 SDK includes the Java compiler. The Sun ONE Studio 4 (also called Forte) is an IDE that includes an editor from which you can compile and run Java programs. Install both pieces of software, and you will have all you need to write and run Java programs.
The CD also contains a trial version of the TextPad IDE as an alternative to the Sun ONE Studio 4 (also called Forte). Both IDEs are good in their own way, and you may, of course, use either one, but our preference is for the TextPad IDE, which is simpler to use and has all you need to work with the material in the text. If you are in a course, take your instructor’s advice on what IDE you should use. If the choice is yours, we suggest that you use the TextPad IDE. Note that in order to use TextPad, you must still install the Java software from Sun described in the previous paragraph.

Macintosh Operating Systems
A recent version of Java for the Mac OS X can be downloaded from the Apple website. When this book went to press, the URL was:

http://developer.apple.com/java/

As we went to press this Mac Java was Java 1.3.1, which should be fine for almost all the code in this book. When the version 1.4.1 Java for the Mac comes out, we suggest that you change to it.

Another good alternative is to purchase a version of CodeWarrior from Metrowerks, Inc. It works well with the Mac operating system.

Linux Operating System
The CD that comes with this text contains a copy of the Java 2 SDK edition, version 1.4, and the Sun ONE Studio 4 (also known as Forte). The Java 2 SDK includes the Java compiler. The Sun ONE Studio 4 (also called Forte) is an IDE that includes an editor from which you can compile and run Java programs. Install these pieces of software, and you will have all you need to write and run Java programs. (Note that, if you prefer, you may use an editor other than the Sun ONE Studio 4 (Forte).)

UNIX Operating Systems (Other than Linux)
If you are in a course using UNIX, then, in all likelihood, Java has already been installed on the course computer for you. If Java has not been installed, visit the Sun Web site to download a suitable version of Java. At the time this book went to press, the URL was

http://java.sun.com/

Self-Test Questions
Each chapter contains numerous self-test questions. Complete answers to all the questions are given at the end of the chapter. One of the best ways to practice what you are learning is to do the self-test questions without looking at the answers. Look at the answers only after you have answered the questions.

This Text Is Also a Reference Book
In addition to using this book as a textbook, you can and should use it as a reference. When you need to check a particular point that you may have forgotten or that you hear mentioned by somebody, but have not yet learned yourself, just look in the index. Many index entries give a page number for “quick reference.” Turn to this quick-reference page. It will contain a short entry, usually set off in a box, that gives all the essential points on that topic. You can consult the quick-reference page to check details of the Java language, as well as details on programming techniques.
Boxed sections in every chapter give you a quick summary of the main points in that
chapter. These boxed sections are labeled “Quick Reference,” “Remember,” and “FAQ.”
You can use these boxes to review the chapter, preview the chapter, or check details of the
Java language.

We Want Your Opinions

This book was written for you, and I would like to hear any comments you have on it. You
can contact me via e-mail at the following address:
wsavitch@ucsd.edu

Unfortunately, I cannot provide you with answers to the programming exercises. Only
instructors who adopt the book can receive (selected) answers from the publisher. For help
with the programming exercises, you will have to contact your instructor. (Even if you are
not enrolled in a class, I still cannot provide answers to programming exercises.) But
remember that there are answers to all the self-test questions at the end of each chapter.

Walter Savitch
http://www.cse.ucsd.edu/users/savitch

Acknowledgments

I thank the Computer Science and Engineering Department of the University of Califor-
nia, San Diego (UCSD), which is my home department and the place that I tested much of
the material covered in this book. Many students in my classes were kind enough to help
correct preliminary versions of the book. These students’ comments and the comments of
instructors who class-tested the book were a tremendous help in shaping the final text. In
particular, I extend a special thanks to Carole McNamee of California State University,
Sacramento, and to Paul Kube of UCSD; their feedback and class-testing of earlier edi-
tions or drafts of the book were a great help to me in producing this edition.

I also thank all the reviewers who took the time to read drafts of this or the previous
editions of the book. They provided invaluable detailed comments and suggestions. In
alphabetical order within each group, they are as follows:

Reviewers for this third edition:
Robert P. Burton—Brigham Young University
Steve Cater—Kettering University
Gobi Gopinath—Suffolk County Community College
Rob Kelly—SUNY, Stony Brook
Michele Kleckner—Elon College
Mike Litman—Western Illinois University
Michael Olan—Stockton State
Ken Slonneger—University of Iowa
Boyd Trolinger—Butte College

In addition, the following individuals were kind enough to provide helpful responses to
a survey done in preparation for this third edition:
Martin Chelten—Moorpark Community College
Tom Cortina—SUNY, Stony Brook
Adel Elmaghraby—University of Louisvile
Gopal Gupta—University of Texas, Dallas
Ricci Heishman—North Virginia Community College
Blayne Mayfield—Oklahoma State University
Subramanian Vijayarangam—University of Massachusetts, Lowell

Reviewers for the second edition:
Jim Buffenbarger—Idaho State University
Martin Chetlen—Moorpark Community College
Tom Cortina—SUNY, Stony Brook
Prasun Dewan—University of North Carolina
Laird Dornan—Sun Microsystems, Inc.
H. E. Dunsmore—Purdue University, Lafayette
Adel Elmaghraby—University of Louisville
Gopal Gupta—New Mexico State University
Le Gruenwald—University of Oklahoma
Ric Heishman—North Virginia Community College
Rob Kelly—SUNY, Stony Brook
Blayne Mayfield—Oklahoma State University
Alan Saleski—Loyola University, Chicago

Reviewers for the first edition:
Michael Clancy—University of California, Berkeley
Michael Godfrey—Cornell University
Robert Herrmann—Sun Microsystems, Inc., Java Soft
Robert Holloway—University of Wisconsin, Madison
Lily Hou—Carnegie Mellon University
John Motil—California State University, Northridge
James Roberts—Carnegie Mellon University
Nan C. Schaller—Rochester Institute of Technology
Ryan Shoemaker—Sun Microsystems, Inc.
Donald E. Smith—Rutgers University

I also thank all the individuals at Prentice Hall who organized the reviewing and production of this book. In particular, I thank Toni Holm for her work in coordinating things between offices and I thank both Lakshmi Balasubramanian and Xiaohong Zhu for their fine job of converting my manuscript into a published book. Thanks also to Patrick Lindner, Heather Scott, and Jake Warde. All these wonderful people cheerfully did a great job. I extend a special thanks to my publisher, Alan Apt, for his invaluable support and advice throughout the writing and production process.

Lew Rakocy prepared the programming solutions for the instructor’s guide, Brian Durney prepared the PowerPoint slides and the test bank questions. I thank both of them for their conscientious work.

I thank Rebecca Pepper for an excellent job of copyediting the manuscript. Since I had final approval on all corrections, any problems that remain in the text are my fault.

I thank Sun Microsystems for allowing me to use the Duke icon in a number of my GUI examples and for permitting us to include Java software on the CD.

Finally, I give a special thanks to Christina for her help and inspiration.

W.S.
Dependency Chart

This chart shows the prerequisites for the chapters in the book. If there is a line between two boxes, the material in the higher box should be covered before the material in the lower box. Minor variations to the chart are discussed in the “Prerequisites” section at the start of each chapter. These variations usually provide more, rather than less, flexibility than what is shown on the chart.

* Note that some sections of these chapters can be covered sooner. Those sections are given in this chart.

** See the chapter’s “Prerequisites” section for full details.

‡ Most of Section 10.1 (“Vectors”) can be covered before covering inheritance.

‡‡ Also requires Section 8.1 of Chapter 8.
BRIEF TABLE OF CONTENTS

Chapter 1  Introduction to Computers and Java 3
Chapter 2  Primitive Types, Strings, and Console I/O 39
Chapter 3  Flow of Control 103
Chapter 4  Defining Classes and Methods 181
Chapter 5  More About Objects and Methods 261
Chapter 6  Arrays 341
Chapter 7  Inheritance 415
Chapter 8  Exception Handling 475
Chapter 9  Streams and File I/O 541
Chapter 10 Dynamic Data Structures 611
Chapter 11 Recursion 669
Chapter 12 Window Interfaces Using Swing 709
Chapter 13 Applets and HTML 795
Chapter 14 More Swing 823
Appendix 1 Keywords 892
Appendix 2 Precedence Rules 893
Appendix 3 Unicode Character Set 894
Appendix 4 SavitchIn 895
Appendix 5 Protected and Package Modifiers 908
Appendix 6 The DecimalFormat Class 909
Appendix 7 The Iterator Interface 912
Appendix 8 Cloning 913
Appendix 9 Javadoc 916
Appendix 10 The JOptionPane Class 918
Appendix 11 Differences between C++ and Java 924

Index 927
# TABLE OF CONTENTS

## CHAPTER 1  Introduction to Computers and Java 3

- Objectives 4
- Prerequisites 4

### 1.1 Computer Basics 4
- Hardware and Memory 5
- Programs 7
- Programming Languages and Compilers 8
- Java Byte-Code 9
- Linking 12

### 1.2 Designing Programs 12
- Object-Oriented Programming 13
- Encapsulation 14
- Polymorphism 15
- Inheritance 16
- If You Know Some Other Programming Language 17
- Algorithms 17
- Reusable Components 18
- Testing and Debugging 18

**Gotcha:** Coping with “Gotchas” 19

**Gotcha:** Hidden Errors 20

### 1.3 A Sip of Java 20
- History of the Java Language 20
- Applets 21
- A First Java Application Program 22
- Compiling a Java Program or Class 28
- Running a Java Program 29
- Preview Examples of Applets (Optional) 30
- Chapter Summary 33
- Answers to Self-Test Questions 34
- Programming Projects 36

## CHAPTER 2  Primitive Types, Strings, and Console I/O 39

- Objectives 40
- Prerequisites 40

### 2.1 Primitive Types and Expressions 40
- Variables 41
- Java Identifiers 44

**Gotcha:** Java Is Case-Sensitive 46

- Primitive Types 46
- Assignment Statements 47
- Specialized Assignment Operators 49
- Simple Input and Output 49
- Number Constants 51
- Assignment Compatibilities 52
- Type Casting 53
Java Tip: Type Casting a Character to an Integer 55
Programming Tip: Initialize Variables 56
Gotcha: Imprecision in Floating-Point Numbers 57
Arithmetic Operators 58
Parentheses and Precedence Rules 60
Case Study: Vending Machine Change 62
Increment and Decrement Operators 66
More about the Increment and Decrement Operators 67

2.2 The Class String 68
String Constants and Variables 68
Concatenation of Strings 69
Classes 70
String Methods 71
String Processing 75
Escape Characters 75
The Unicode Character Set 77

2.3 Keyboard and Screen I/O 79
Screen Output 79
Input Using SavitchIn 82
More Input Methods 85
Gotcha: readInt and readDouble 86
Programming Tip: Echo Input 87

2.4 Documentation and Style 88
Programming Tip: Use Meaningful Names for Variables 88
Documentation and Comments 89
Indenting 90
Named Constants 92
Chapter Summary 95
Answers to Self-Test Questions 96
Programming Projects 99

CHAPTER 3  Flow of Control 103
Objectives 104
Prerequisites 104

3.1 Branching Statements 104
The if-else Statement 104
Introduction to Boolean Expressions 108
Gotcha: Using == with Strings 111
Programming Tip: Alphabetical Order 113
Nested Statements and Compound Statements 115
Java Tip: Matching else and if 117
Multibranch if-else Statements 118
Programming Example: Assigning Letter Grades 119
The switch Statement 122
Gotcha: Omitting a break Statement 126
The Conditional Operator (Optional) 126
3.2 Java Loop Statements 128
while Statements 129
Java Tip: A while Loop Can Perform Zero Iterations 133
The do-while Statement 133
Programming Example: Bug Infestation 137
Gotcha: Infinite Loops 139
The for Statement 141
The Comma in for Statements (Optional) 145
Gotcha: Extra Semicolon in a Loop Statement 146
Java Tip: Choosing a Loop Statement 148
The break Statement in Loops 148
Gotcha: Misuse of break Statements 150
The exit Method 150

3.3 Programming with Loops 152
The Loop Body 152
Initializing Statements 154
Ending a Loop 155
Programming Example: Nested Loops 156
Programming Tip: Avoid Declaring Variables in a Loop Body 157
Loop Bugs 157
Tracing Variables 161

3.4 The Type boolean 162
Boolean Expressions and Boolean Variables 163
Programming Tip: Naming Boolean Variables 164
Precedence Rules 165
Input and Output of Boolean Values 168
Case Study: Using a Boolean Variable to End a Loop 168
Chapter Summary 172
Answers to Self-Test Questions 172
Programming Projects 177

CHAPTER 4 Defining Classes and Methods 181
Objectives 182
Prerequisites 182

4.1 Class and Method Definitions 182
Class Files and Separate Compilation 184
Instance Variables 185
Using Methods 188
void Method Definitions 191
Methods That Return a Value 193
Java Tip: Use of return in void Methods 196
The this Parameter 198
Local Variables 200
Blocks 202
Gotcha: Variables Declared in a Block 202
Java Tip: Declaring Variables in a for Statement 203
Parameters of a Primitive Type 203

Gotcha: Use of the Terms Parameter and Argument 209

Summary of Class and Method Definition Syntax 209

4.2 Information Hiding and Encapsulation 210

Information Hiding 210

Programming Tip: Parameter Names Are Local to the Method 211

Precondition and Postcondition Comments 211

Java Tip: Assertion Checks 212

The public and private Modifiers 214

Programming Tip: Instance Variables Should Be private 217

Programming Example: A Purchase Class 221

Encapsulation 225

Automatic Documentation with javadoc 228

UML Class Diagrams 228

4.3 Objects and Reference 230

Variables of a Class Type and Objects 230

Gotcha: Use of = and == with Variables of a Class Type 235

Java Tip: Define an equals Method for Your Classes 235

Programming Example: A Species Class 239

Boolean-Valued Methods 242

Class Parameters 244

Comparing Class Parameters and Primitive-Type Parameters 247

Chapter Summary 251

Answers to Self-Test Questions 251

Programming Projects 257

CHAPTER 5 More About Objects and Methods 261

Objectives 262

Prerequisites 262

5.1 Programming with Methods 263

Methods Calling Methods 263

Programming Tip: Make Helping Methods Private 268

Java Tip: Make the Compiler Happy 268

Gotcha: Null Pointer Exception 269

5.2 Static Methods and Static Variables 271

Static Methods 271

Gotcha: Invoking a Nonstatic Method Within a Static Method 273

Java Tip: You Can Put a main in Any Class 275

Static Variables (Optional) 277

The Math Class 279

Integer, Double, and Other Wrapper Classes 282

5.3 Designing Methods 286

Case Study: Formatting Output 287

Top-Down Design 291

Testing Methods 293
Searching an Array 380

Gotcha: Returning an Array Instance Variable 381

6.4 Sorting Arrays 383
Selection Sort 383
Other Sorting Algorithms 388

6.5 Multidimensional Arrays 388
Multidimensional-Array Basics 390
Multidimensional-Array Parameters and Returned Values 392
Implementation of Multidimensional Arrays 394
Ragged Arrays (Optional) 397
Programming Example: Employee Time Records 398
Chapter Summary 404
Answers to Self-Test Questions 405
Programming Projects 410

CHAPTER 7 Inheritance 415
Objectives 416
Prerequisites 416

7.1 Inheritance Basics 416
Programming Example: A Person Class 417
Derived Classes 418
Overriding Method Definitions 421
Overriding Versus Overloading 422
The final Modifier 422
Gotcha: Use of Private Instance Variables from the Base Class 423
Programming Tip: Assume That Your Coworkers Are Malicious 424
Gotcha: Private Methods Are Not Inherited 424
UML Inheritance Diagrams 424

7.2 Programming with Inheritance 427
Constructors in Derived Classes 427
The this Method (Optional) 428
Call to an Overridden Method 429
Programming Example: Multilevel Derived Classes 430
A Subtle Point About Overloading and Overriding (Optional) 435
Java Tip: You Cannot Use Multiple supers 435
Programming Tip: An Object Can Have More than One Type 436
Programming Tip: “Is a” and “Has a” Relations 439

The Class Object 440
Case Study: Character Graphics 442
Abstract Classes 452
Interfaces (Optional) 456

7.3 Dynamic Binding and Polymorphism 458
Dynamic Binding 458
Type Checking and Dynamic Binding 460
Dynamic Binding with toString 461
CHAPTER 8 Exception Handling 475
Objectives 476
Prerequisites 476

8.1 Basic Exception Handling 477
Exceptions in Java 477
Predefined Exception Classes 489
ArrayIndexOutOfBoundsException (Alternative Ordering) 490

8.2 Defining Exception Classes 491
Defining Your Own Exception Classes 491
Java Tip: Preserve getMessage When You Define Exception Classes 496
Programming Tip: When to Define an Exception Class 497

8.3 Using Exception Classes 502
Declaring Exceptions (Passing the Buck) 502
Exceptions That Do Not Need To Be Caught 506
The AssertionError Class (Optional) 507
Multiple Throws and Catches 507
Java Tip: Catch the More Specific Exception First 510
Programming Tip: Exception Handling and Information Hiding 511
Gotcha: Overuse of Exceptions 513
Programming Tip: When to Throw an Exception 513
Gotcha: Nested try-catch Blocks 515
The finally Block (Optional) 515
Rethrowing an Exception (Optional) 516
Case Study: A Line-Oriented Calculator 517
Chapter Summary 531
Answers to Self-Test Questions 532
Programming Projects 536

CHAPTER 9 Streams and File I/O 541
Objectives 542
Prerequisites 542

9.1 An Overview of Streams and File I/O 543
The Concept of a Stream 543
Why Use Files for I/O? 543
Differences between Text Files and Binary Files 543

9.2 Text-File I/O 545
Text-File Output with PrintWriter 545
Gotcha: A try Block Is a Block 550
Gotcha: Overwriting a File 551
<table>
<thead>
<tr>
<th>Java Tip: Append To a Text File</th>
<th>551</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Tip: Use toString for Text-File Output</td>
<td>553</td>
</tr>
<tr>
<td>Text-File Input with BufferedReader</td>
<td>556</td>
</tr>
<tr>
<td>Programming Example: Reading a File Name from the Keyboard</td>
<td>560</td>
</tr>
<tr>
<td>Java Tip: Using Path Names</td>
<td>562</td>
</tr>
<tr>
<td>The StringTokenizer Class</td>
<td>563</td>
</tr>
<tr>
<td>Java Tip: Testing for the End of a Text File</td>
<td>565</td>
</tr>
<tr>
<td>The Classes FileReader and FileOutputStream</td>
<td>567</td>
</tr>
<tr>
<td>Unwrapping the Class SavitchIn</td>
<td>569</td>
</tr>
</tbody>
</table>

**9.3 The File Class 570**

Using the File Class 570

**9.4 Basic Binary-File I/O 573**

Output to Binary Files, Using ObjectOutputStream 574

Some Details about writeUTF (Optional) 579

Reading Input from a Binary File, Using ObjectInputStream 580

Gotcha: Using ObjectInputStream with a Text File 584

Gotcha: Defining a Method to Open a Stream 585

The EOFException Class 586

Gotcha: Forgetting to Check for the End of a File 588

Gotcha: Checking for the End of a File in the Wrong Way 589

The Classes FileInputStream and FileOutputStream 589

Programming Example: Processing a File of Binary Data 590

**9.5 Object I/O with Object Streams 594**

Binary I/O of Class Objects 594

Gotcha: Exceptions, Exceptions, Exceptions 599

The Serializable Interface 599

Gotcha: Mixing Class Types in the Same File 599

Array Objects in Binary Files 600

Chapter Summary 602

Answers to Self-Test Questions 603

Programming Projects 608

**CHAPTER 10 Dynamic Data Structures 611**

Objectives 612

Prerequisites 612

**10.1 Vectors 613**

Using Vectors 613

Programming Tip: Adding to a Vector 619

Gotcha: Vector Elements Are of Type Object 620

Comparing Vectors and Arrays 622

Gotcha: Using capacity Instead of size 622

Java Tip: Use trimToSize to Save Memory 624

Gotcha: Using the Method clone 625

Java Tip: Newer Collection Classes (Optional) 626
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.2</td>
<td>Linked Data Structures</td>
<td>627</td>
</tr>
<tr>
<td></td>
<td>Linked Lists</td>
<td>627</td>
</tr>
<tr>
<td></td>
<td>Gotcha: Null Pointer Exception</td>
<td>636</td>
</tr>
<tr>
<td></td>
<td>Gotcha: Privacy Leaks</td>
<td>637</td>
</tr>
<tr>
<td></td>
<td>Inner Classes</td>
<td>638</td>
</tr>
<tr>
<td></td>
<td>Node Inner Classes</td>
<td>639</td>
</tr>
<tr>
<td></td>
<td>Iterators</td>
<td>639</td>
</tr>
<tr>
<td></td>
<td>Programming Tip: Internal and External Iterators</td>
<td>652</td>
</tr>
<tr>
<td></td>
<td>Exception Handling with Linked Lists</td>
<td>652</td>
</tr>
<tr>
<td></td>
<td>Variations on a Linked List</td>
<td>656</td>
</tr>
<tr>
<td></td>
<td>Other Linked Data Structures</td>
<td>658</td>
</tr>
<tr>
<td></td>
<td>Chapter Summary</td>
<td>658</td>
</tr>
<tr>
<td></td>
<td>Answers to Self-Test Questions</td>
<td>659</td>
</tr>
<tr>
<td></td>
<td>Programming Projects</td>
<td>665</td>
</tr>
<tr>
<td>Chapter 11</td>
<td>Recursion</td>
<td>669</td>
</tr>
<tr>
<td></td>
<td>Objectives</td>
<td>670</td>
</tr>
<tr>
<td></td>
<td>Prerequisites</td>
<td>670</td>
</tr>
<tr>
<td>11.1</td>
<td>The Basics of Recursion</td>
<td>670</td>
</tr>
<tr>
<td></td>
<td>Case Study: Digits to Words</td>
<td>671</td>
</tr>
<tr>
<td></td>
<td>How Recursion Works</td>
<td>675</td>
</tr>
<tr>
<td></td>
<td>Gotcha: Infinite Recursion</td>
<td>679</td>
</tr>
<tr>
<td></td>
<td>Recursive versus Iterative Definitions</td>
<td>682</td>
</tr>
<tr>
<td></td>
<td>Recursive Methods That Return a Value</td>
<td>682</td>
</tr>
<tr>
<td>11.2</td>
<td>Programming with Recursion</td>
<td>687</td>
</tr>
<tr>
<td></td>
<td>Programming Tip: Ask Until the User Gets It Right</td>
<td>687</td>
</tr>
<tr>
<td></td>
<td>Case Study: Binary Search</td>
<td>689</td>
</tr>
<tr>
<td></td>
<td>Programming Tip: Generalize the Problem</td>
<td>693</td>
</tr>
<tr>
<td></td>
<td>Programming Example: Merge Sort—A Recursive Sorting Method</td>
<td>697</td>
</tr>
<tr>
<td></td>
<td>Chapter Summary</td>
<td>702</td>
</tr>
<tr>
<td></td>
<td>Answers to Self-Test Questions</td>
<td>702</td>
</tr>
<tr>
<td></td>
<td>Programming Projects</td>
<td>705</td>
</tr>
<tr>
<td>Chapter 12</td>
<td>Window Interfaces Using Swing</td>
<td>709</td>
</tr>
<tr>
<td></td>
<td>Objectives</td>
<td>710</td>
</tr>
<tr>
<td></td>
<td>Prerequisites</td>
<td>710</td>
</tr>
<tr>
<td>12.1</td>
<td>Background</td>
<td>711</td>
</tr>
<tr>
<td></td>
<td>GUIs—Graphical User Interfaces</td>
<td>711</td>
</tr>
<tr>
<td></td>
<td>Event-Driven Programming</td>
<td>711</td>
</tr>
<tr>
<td>12.2</td>
<td>Basic Swing Details</td>
<td>713</td>
</tr>
<tr>
<td></td>
<td>Gotcha: Save All Your Work before Running a Swing Program</td>
<td>714</td>
</tr>
<tr>
<td></td>
<td>Programming Example: A Simple Window</td>
<td>714</td>
</tr>
<tr>
<td></td>
<td>Java Tip: Ending a Swing Program</td>
<td>719</td>
</tr>
<tr>
<td></td>
<td>Gotcha: Forgetting to Program the Close-Window Button</td>
<td>720</td>
</tr>
<tr>
<td></td>
<td>Gotcha: Forgetting to Use getContentPane</td>
<td>720</td>
</tr>
</tbody>
</table>
Table of Contents  xxv

More about Window Listeners 720
Size Units for Screen Objects 722
More on setVisible 723
Programming Example: A Better Version of Our First Swing Program 725
Programming Example: A Window with Color 728
Some Methods of the Class JFrame 732
Layout Managers 735

12.3  Buttons and Action Listeners 742
Programming Example: Adding Buttons 742
Buttons 744
Action Listeners and Action Events 746
Gotcha: Changing the Parameter List for actionPerformed 750
Interfaces 751
Java Tip: Code a GUI’s Look and Actions Separately 752
The Model–View–Controller Pattern 753
Java Tip: Use the Method setActionCommand 753

12.4  Container Classes 756
The JPanel Class 756
The Container Class 759
Java Tip: Guide for Creating Simple Window Interfaces 762

12.5  Text I/O for GUIs 764
Text Areas and Text Fields 764
Programming Example: Labeling a Text Field 770
Inputting and Outputting Numbers 772
Programming Example: A GUI Adding Machine 775
Catching a NumberFormatException 779
Chapter Summary 782
Answers to Self-Test Questions 783
Programming Projects 791

CHAPTER 13  Applets and HTML 795
Objectives 796
Prerequisites 796

13.1  Applets 797
Applet Basics 797
Running an Applet 800
Programming Example: An Adder Applet 800
Java Tip: Converting a Swing Application to an Applet 803
Adding Icons to an Applet 803

13.2  Introduction to HTML 806
HTML Basics 806
Programming Tip: A Simple HTML-Document Outline 808
Inserting Hyperlinks 808
Gotcha: Not Using Your Reload (Refresh) Button 812
xxvi  Table of Contents

Displaying a Picture 813
Programming Tip:  HTML Is a Low-Level Language 814

13.3 Applets in HTML 814
Placing an Applet in an HTML Document 814
Java Tip:  Applet Viewer Revisited 816
Gotcha:  Using an Old Web Browser 817
The Older Applet Class (Optional) 817
Applets and Security 818
Chapter Summary 819
Answers to Self-Test Questions 819
Programming Projects 820

CHAPTER 14 More Swing 823
Objectives 824
Prerequisites 824

14.1 Menus 824
Programming Example: A GUI with a Menu 825
Menu Bars, Menus, and Menu Items 825
Nested Menus 830

14.2 Making GUIs Pretty (and More Functional) 832
Adding Icons 832
Gotcha: Resizing Buttons 838
The JScrollPane Class for Scroll Bars 839
Adding Borders 845
Gotcha: Forgetting to Import javax.swing.border 850

14.3 More Layout Managers 851
TheBoxLayout Manager Class 851
Struts and Glue 855
Gotcha: Using Struts and Glue with Other Layout Managers 856
Setting the Spacing between Components 857
The BoxContainer Class 857
The CardLayout Manager 860

14.4 Inner Classes 866
Helping Classes 866

14.5 More on Events and Listeners 868
The WindowListener Interface 868
Java Tip: Programming the Close-Window Button 873
Java Tip: More about setDefaultCloseOperation 877
Programming Example: Components with Changing Visibility 879
Some More Details on Updating a GUI 882

14.6 Another Look at the Swing Class Hierarchy 883
Buttons, Menus, and Abstract Buttons 883
Java Tip: More Constructors for the Class JMenuItem 884
Java Tip: There Are a Lot More Swing Classes and Methods 885
Chapter Summary 886
Answers to Self-Test Questions 886
Programming Projects 890
Appendix 1  Keywords 892
Appendix 2  Precedence Rules 893
Appendix 3  Unicode Character Set 894
Appendix 4  SavitchIn 895
Appendix 5  Protected and Package Modifiers 908
Appendix 6  The DecimalFormat Class 909
  Percent Notation 909
  Scientific Notation (E-Notation) 911
Appendix 7  The Iterator Interface 912
Appendix 8  Cloning 913
Appendix 9  Javadoc 916
  Commenting Classes for Use with javadoc 916
  Running javadoc 917
Appendix 10  The JOptionPane Class 918
  A Sample Program Using JOptionPane 918
  Multiline Output Windows 923
Appendix 11  Differences between C++ and Java 924
  Primitive Types 924
  Strings 924
  Flow of Control 924
  Testing for Equality 925
  main Method (Function) and Other Methods 925
  Files and Including Files 925
  Class and Method (Function) Definitions 925
  No Pointers in Java 925
  Method (Function) Parameters 926
  Arrays 926
  Garbage Collection 926
  Other Comparisons 926
Index 927