Preface

Welcome to the Seventh Edition of *Java Software Solutions: Foundations of Program Design*. We are pleased that this book has served the needs of so many students and faculty over the years. This edition has been tailored further to improve the coverage of topics key to introductory computing.

**New to This Edition**

- Split Chapter 5 of the 6th edition into two for better coverage and flow.
- Moved the coverage of the `ArrayList` class earlier in the book to permit more interesting projects earlier.
- Improved the discussion of an array as a programming construct.
- Improved the discussions of visibility modifiers, especially regarding the `protected` modifier.
- Replaced and updated examples throughout the book.
- Replaced, updated, and added exercises and programming projects.
- Available with MyProgrammingLab (see details later in this Preface).

Feedback from both instructors and students continues to make it clear that we have hit the mark with the overall vision of the book. The emphasis remains on presenting underlying core concepts in a clear and gradual manner. The Graphics Track sections in each chapter still segregate the coverage of graphics and graphical user interfaces, giving extreme flexibility in how that material gets covered. The casual writing style and entertaining examples still rule the day.

The enhancements in this edition are designed to allow the instructor more flexibility in topic coverage. In an attempt to cover all issues related to conditionals and loops, Chapter 5 in the previous edition had become very large and a bit too encyclopedic. In this edition that chapter has been carefully redesigned into two, giving the coverage of those topics a better flow. The new organization allows more interesting examples to be explored earlier.

One effect of this reorganization is that it allowed us to bring the coverage of the `ArrayList` class earlier in the book. Although arrays are used internally to
implement the `ArrayList` class, there is no reason to wait for arrays to be covered to introduce the `ArrayList` class. Like many other classes in the Java API, the `ArrayList` class can be used without needing to know how it works internally. An `ArrayList` object can be used for its (very valuable) functionality as soon as loops are available. The new organization in this edition does exactly that. If the instructor chooses, coverage of `ArrayList` can still be deferred as it has been before, but now the option is there to introduce them earlier.

In addition to these changes, various discussions throughout the book have been revamped and improved. For example, the explanation of the effects of the `protected` visibility modifier has enhanced to clarify its use. Furthermore, throughout the book older examples have been rejuvenated, and end-of-chapter exercises and programming projects have been augmented.

Cornerstones of the Text

This text is based on the following basic ideas that we believe make for a sound introductory text:

- **True object-orientation.** A text that really teaches a solid object-oriented approach must use what we call object-speak. That is, all processing should be discussed in object-oriented terms. That does not mean, however, that the first program a student sees must discuss the writing of multiple classes and methods. A student should learn to use objects before learning to write them. This text uses a natural progression that culminates in the ability to design real object-oriented solutions.

- **Sound programming practices.** Students should not be taught how to program; they should be taught how to write good software. There’s a difference. Writing software is not a set of cookbook actions, and a good program is more than a collection of statements. This text integrates practices that serve as the foundation of good programming skills. These practices are used in all examples and are reinforced in the discussions. Students learn how to solve problems as well as how to implement solutions. We introduce and integrate basic software engineering techniques throughout the text. The **Software Failure** vignettes reiterate these lessons by demonstrating the perils of not following these sound practices.

- **Examples.** Students learn by example. This text is filled with fully implemented examples that demonstrate specific concepts. We have intertwined small, readily understandable examples with larger, more realistic ones. There is a balance between graphics and nongraphics programs. The **VideoNotes** provide additional examples in a live presentation format.
■ Graphics and GUIs. Graphics can be a great motivator for students, and their use can serve as excellent examples of object-orientation. As such, we use them throughout the text in a well-defined set of sections that we call the Graphics Track. This coverage includes the use of event processing and GUIs. Students learn to build GUIs in the appropriate way by using a natural progression of topics. The Graphics Track can be avoided entirely for those who do not choose to use graphics.

Chapter Breakdown

Chapter 1 (Introduction) introduces computer systems in general, including basic architecture and hardware, networking, programming, and language translation. Java is introduced in this chapter, and the basics of general program development, as well as object-oriented programming, are discussed. This chapter contains broad introductory material that can be covered while students become familiar with their development environment.

Chapter 2 (Data and Expressions) explores some of the basic types of data used in a Java program and the use of expressions to perform calculations. It discusses the conversion of data from one type to another and how to read input interactively from the user with the help of the standard Scanner class.

Chapter 3 (Using Classes and Objects) explores the use of predefined classes and the objects that can be created from them. Classes and objects are used to manipulate character strings, produce random numbers, perform complex calculations, and format output. Enumerated types are also discussed.

Chapter 4 (Writing Classes) explores the basic issues related to writing classes and methods. Topics include instance data, visibility, scope, method parameters, and return types. Encapsulation and constructors are covered as well. Some of the more involved topics are deferred to or revisited in Chapter 6.

Chapter 5 (Conditionals and Loops) covers the use of boolean expressions to make decisions. Then the if statement and while loop are explored in detail. Once loops are established, the concept of an iterator is introduced and the Scanner class is revisited for additional input parsing and the reading of text files. Finally, the ArrayList class introduced, which provides the option for managing a large number of objects.

Chapter 6 (More Conditionals and Loops) examines the rest of Java’s conditional (switch) and loop (do, for) statements. All related statements for conditionals and loops are discussed, including the enhanced version of the for loop. The for-each loop is also used to process iterators and ArrayList objects.
Chapter 7 (Object-Oriented Design) reinforces and extends the coverage of issues related to the design of classes. Techniques for identifying the classes and objects needed for a problem and the relationships among them are discussed. This chapter also covers static class members, interfaces, and the design of enumerated type classes. Method design issues and method overloading are also discussed.

Chapter 8 (Arrays) contains extensive coverage of arrays and array processing. The nature of an array as a low-level programming structure is contrasted to the higher-level object management approach. Additional topics include command-line arguments, variable length parameter lists, and multidimensional arrays.

Chapter 9 (Inheritance) covers class derivations and associated concepts such as class hierarchies, overriding, and visibility. Strong emphasis is put on the proper use of inheritance and its role in software design.

Chapter 10 (Polymorphism) explores the concept of binding and how it relates to polymorphism. Then we examine how polymorphic references can be accomplished using either inheritance or interfaces. Sorting is used as an example of polymorphism. Design issues related to polymorphism are examined as well.

Chapter 11 (Exceptions) explores the class hierarchy from the Java standard library used to define exceptions, as well as the ability to define our own exception objects. We also discuss the use of exceptions when dealing with input and output and examine an example that writes a text file.

Chapter 12 (Recursion) covers the concept, implementation, and proper use of recursion. Several examples from various domains are used to demonstrate how recursive techniques make certain types of processing elegant.

Chapter 13 (Collections) introduces the idea of a collection and its underlying data structure. Abstraction is revisited in this context and the classic data structures are explored. Generic types are introduced as well. This chapter serves as an introduction to a CS2 course.

Supplements

Student Online Resources
These student resources can be accessed at the book’s Companion Website, www.pearsonhighered.com/lewis:

- Source Code for all the programs in the text
- Links to Java development environments
- VideoNotes: short step-by-step videos demonstrating how to solve problems from design through coding. VideoNotes allow for self-paced
Online Practice and Assessment

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

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

MyProgrammingLab is offered to users of this book in partnership with Turing’s Craft, the makers of the CodeLab interactive programming exercise system. For a full demonstration, to see feedback from instructors and students, or to get started using MyProgrammingLab in your course, visit www.myprogramminglab.com.

Instructor Resources

The following supplements are available to qualified instructors only. Visit the Pearson Education Instructor Resource Center (www.pearsonhighered.com/irc) or send an e-mail to computing@pearson.com for information on how to access them:

- Presentation Slides—in PowerPoint.
- Solutions—includes solutions to exercises and programming projects.
- Test Bank with powerful test generator software—includes a wealth of free response, multiple-choice, and true/false type questions.
- Lab Manual—lab exercises are designed to accompany the topic progression in the text.
Java Integrated Development Environment (IDE) Resource Kits

Instructors can order this text with a kit that includes a disk containing 7 popular Java IDEs (the most recent JDK from Oracle, Eclipse, NetBeans, jGRASP, DrJava, BlueJ, and TextPad) and access to a website containing written and video tutorials for getting started in each IDE. For Instructors, ordering information can be found at www.pearsonhighered.com/cs, or from your campus Pearson Education sales representative. For Students, if your instructor didn’t request the Java IDE Resource Kit, links for downloading the IDEs can be found at the book’s Companion Website.

Features

Key Concepts. Throughout the text, the Key Concept boxes highlight fundamental ideas and important guidelines. These concepts are summarized at the end of each chapter.

Listings. All programming examples are presented in clearly labeled listings, followed by the program output, a sample run, or screen shot display as appropriate. The code is colored to visually distinguish comments and reserved words.

Syntax Diagrams. At appropriate points in the text, syntactic elements of the Java language are discussed in special highlighted sections with diagrams that clearly identify the valid forms for a statement or construct. Syntax diagrams for the entire Java language are presented in Appendix L.

Graphics Track. All processing that involves graphics and graphical user interfaces is discussed in one or two sections at the end of each chapter that we collectively refer to as the Graphics Track. This material can be skipped without loss of continuity, or focused on specifically as desired. The material in any Graphics Track section relates to the main topics of the chapter in which it is found. Graphics Track sections are indicated by a brown border on the edge of the page.

Summary of Key Concepts. The Key Concepts presented throughout a chapter are summarized at the end of the chapter.

Self-Review Questions and Answers. These short-answer questions review the fundamental ideas and terms established in the preceding section. They are designed to allow students to assess their own basic grasp of the material. The answers to these questions can be found at the end of the book in Appendix N.

Exercises. These intermediate problems require computations, the analysis or writing of code fragments, and a thorough grasp of the chapter content. While the exercises may deal with code, they generally do not require any online activity.
**Programming Projects.** These problems require the design and implementation of Java programs. They vary widely in level of difficulty.

**MyProgrammingLab.** Many of the problems in the book can be done online in MyProgrammingLab. Through practice exercises and immediate, personalized feedback, MyProgrammingLab improves the programming competence of beginning students who often struggle with the basic concepts and paradigms of popular high-level programming languages.

**VideoNotes.** Presented by the author, VideoNotes explain topics visually through informal videos in an easy-to-follow format, giving students the extra help they need to grasp important concepts. Look for this VideoNote icon to see which in-chapter topics and end-of-chapter Programming Projects are available as VideoNotes.

**Software Failures.** These between-chapter vignettes discuss real-world flaws in software design, encouraging students to adopt sound design practices from the beginning.

**Acknowledgments**

I am most grateful to the faculty and students from around the world who have provided their feedback on previous editions of this book. I am pleased to see the depth of the faculty’s concern for their students and the students’ thirst for knowledge. Your comments and questions are always welcome.

I am particularly thankful for the assistance, insight, and attention to detail of Robert Burton from Brigham Young University. For years, Robert has consistently provided valuable feedback that helps shape and evolve this textbook. Recently he also performed a revision of the material in Chapter 1 about personal computing systems that brought it back to a state-of-the-art discussion.

Brian Fraser of Simon Fraser University also has recently provided some excellent feedback that helped clarify some issues in this edition. Such interaction with computing educators is incredibly valuable.

I also want to thank Dan Joyce from Villanova University, who developed the Self-Review questions, ensuring that each relevant topic had enough review material, as well as developing the answers to each.

I continue to be amazed at the talent and effort demonstrated by the team at Pearson Addison-Wesley. Michael Hirsch, our editor, has amazing insight and commitment. His assistant, Stephanie Sellinger, is a source of consistent and helpful support. Marketing Manager Yez Alayan makes sure that instructors understand the pedagogical advantages of the text. The cover was designed by the skilled talents of Suzanne Harbison. Jeff Holcomb and Heather McNally led the production effort.
The Addison-Wesley folks were supported by a phenomenal team at Nesbitt Graphics, including Jerilyn Bockorick for the interior design, Rose Kernan for project management, Diane Paluba for production coordination. We thank all of these people for ensuring that this book meets the highest quality standards.

Special thanks go to the following people who provided valuable advice to us about this book via their participation in focus groups, interviews, and reviews. They, as well as many other instructors and friends, have provided valuable feedback. They include:

Elizabeth Adams
David Atkins
Lewis Barnett
Thomas W. Bennet
Gian Mario Besana
Hans-Peter Bischof
Robert Burton
John Chandler
Robert Cohen
Dodi Coreson
James H. Cross II
Eman El-Sheikh
Christopher Eliot
Wanda M. Eanes
Stephanie Elzer
Matt Evett
Marj Feroe
John Gauch
Chris Haynes
James Heliotis
Laurie Hendren
Mike Higgs
Stephen Hughes
Saroja Kanchi
Karen Kluge
Jason Levy
Peter MacKenzie
Blayne Mayfield
Gheorghe Muresan
Laurie Murphy
Dave Musicant
Faye Navabi-Tadayon

James Madison University
University of Oregon
University of Richmond
Mississippi College
DePaul University
Rochester Institute of Technology
Brigham Young University
Oklahoma State University
University of Massachusetts, Boston
Linn Benton Community College
Auburn University
University of West Florida
University of Massachusetts, Amherst
Macon State College
Millersville University
Eastern Michigan University
Delaware County Community College, Pennsylvania
University of Kansas
Indiana University
Rochester Institute of Technology
McGill University
Austin College
Roanoke College
Kettering University
Dartmouth College
University of Hawaii
McGill University
Oklahoma State University
Rutgers University
Pacific Lutheran University
Carleton College
Arizona State University
Thanks also go to my friends and former colleagues at Villanova University who have provided so much wonderful feedback. They include Bob Beck, Cathy Helwig, Anany Levitin, Najib Nadi, Beth Taddei, and Barbara Zimmerman.

Special thanks go to Pete DePasquale of The College of New Jersey for the design and evolution of the PaintBox project, as well as the original Java Class Library appendix.

Many other people have helped in various ways. They include Ken Arnold, Mike Czepiel, John Loftus, Sebastian Niezgoda, and Saverio Perugini. Our apologies to anyone we may have omitted.

The ACM Special Interest Group on Computer Science Education (SIGCSE) is a tremendous resource. Their conferences provide an opportunity for educators from all levels and all types of schools to share ideas and materials. If you are an educator in any area of computing and are not involved with SIGCSE, you’re missing out.