

LEWIS • LOFTUS



*Java*TM
SOFTWARE SOLUTIONS

FOUNDATIONS OF
PROGRAM DESIGN

Seventh Edition

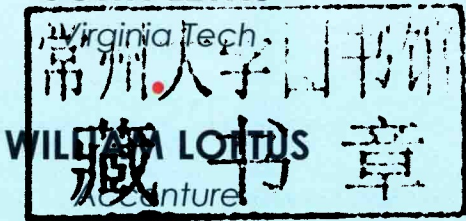
Seventh Edition

Java™

SOFTWARE SOLUTIONS

FOUNDATIONS OF PROGRAM DESIGN

JOHN LEWIS



Addison-Wesley

Boston Columbus Indianapolis New York San Francisco Upper Saddle River
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal Toronto
Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Editorial Director:	Marcia Horton	Image Permission Coordinator:	Rita Wenning
Editor-in-Chief:	Michael Hirsch	Cover Photograph:	© Creative Crop/Digital Vision/Getty Images
Editorial Assistant:	Stephanie Sellinger	Media Editor:	Daniel Sandin
Vice President, Marketing:	Patrice Jones	Media Project Manager:	Wanda Rockwell
Marketing Manager:	Yezan Alayan	Full-Service Project Management:	Rose Kernan, Nesbitt Graphics, Inc.
Marketing Coordinator:	Kathryn Ferranti	Composition:	Glyph International
Vice President, Production:	Vince O'Brien	Printer/Binder:	Quad/Graphics Book Services, Taunton
Managing Editor:	Jeff Holcomb	Cover Printer:	Coral Graphics Services, Inc.
Production Project Manager:	Heather McNally	Text Font:	Sabon LT Std
Senior Operations Supervisor:	Alan Fischer		
Manufacturing Buyer:	Lisa McDowell		
Art Director:	Linda Knowles		
Cover Designer:	Suzanne Harbison		

Credits and acknowledgments borrowed from other sources and reproduced, with permission, in this textbook appear below, or on appropriate page within text.

Photo Credits: Page 11: NASA Earth Observing System. Page 205: Susan Van Etten /PhotoEdit. Page 267: David Joel /Stone/Getty Images. Page 377 (left and right): National Oceanic and Atmospheric Administration NOAA. Page 441: Matthew McVay/Stone/Getty Images. Page 485: Mario Fourmy/REA/Redux Pictures.

Microsoft® and Windows® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. Screen shots and icons reprinted with permission from the Microsoft Corporation. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

Copyright © 2012, 2009, 2007, 2005, 2003 Pearson Education, Inc., publishing as Addison-Wesley, 501 Boylston Street, Suite 900, Boston, Massachusetts 02116. All rights reserved. Manufactured in the United States of America. This publication is protected by Copyright, and permission should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission(s) to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, Addison-Wesley, 501 Boylston Street, Suite 900, Boston, Massachusetts 02116.

Many of the designations by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed in initial caps or all caps.

Library of Congress Cataloging-in-Publication Data

Lewis, John, 1963-

Java software solutions : foundations of program design / John Lewis & William Loftus.

-- 7th ed.

p. cm.

Includes bibliographical references and index.

ISBN 978-0-13-214918-1 (alk. paper)

1. Java (Computer program language) 2. Object-oriented programming (Computer science) I. Loftus, William. II. Title.

QA76.73.J38L49 2012

005.13'3--dc22

2011001726

10 9 8 7 6 5 4 3 2 1—QGT—15 14 13 12 11

Addison-Wesley
is an imprint of

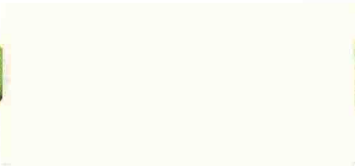
PEARSON

ISBN 10: 0-13-214918-4
ISBN 13: 978-0-13-214918-1



get with the programming

Through the power of practice and immediate personalized feedback, MyProgrammingLab improves your performance.™

PEARSON
mypro  **ninglab**™

Learn more at www.myprogramminglab.com

*This book is dedicated to our families.
Sharon, Justin, Kayla, Nathan, and Samantha Lewis
and
Veena, Isaac, and Dévi Loftus*

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 is 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

instruction with easy navigation including the ability to select, play, re-wind, fast-forward, and stop within each VideoNote exercise. Margin icons in your textbook let you know when a VideoNote video is available for a particular concept or homework problem.

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	James Madison University
David Atkins	University of Oregon
Lewis Barnett	University of Richmond
Thomas W. Bennet	Mississippi College
Gian Mario Besana	DePaul University
Hans-Peter Bischof	Rochester Institute of Technology
Robert Burton	Brigham Young University
John Chandler	Oklahoma State University
Robert Cohen	University of Massachusetts, Boston
Dodi Coreson	Linn Benton Community College
James H. Cross II	Auburn University
Eman El-Sheikh	University of West Florida
Christopher Eliot	University of Massachusetts, Amherst
Wanda M. Eanes	Macon State College
Stephanie Elzer	Millersville University
Matt Evett	Eastern Michigan University
Marj Feroe	Delaware County Community College, Pennsylvania
John Gauch	University of Kansas
Chris Haynes	Indiana University
James Heliotis	Rochester Institute of Technology
Laurie Hendren	McGill University
Mike Higgs	Austin College
Stephen Hughes	Roanoke College
Saroja Kanchi	Kettering University
Karen Kluge	Dartmouth College
Jason Levy	University of Hawaii
Peter MacKenzie	McGill University
Blayne Mayfield	Oklahoma State University
Gheorghe Muresan	Rutgers University
Laurie Murphy	Pacific Lutheran University
Dave Musicant	Carleton College
Faye Navabi-Tadayon	Arizona State University

Lawrence Osborne	Lamar University
Barry Pollack	City College of San Francisco
B. Ravikumar	University of Rhode Island
David Riley	University of Wisconsin (La Crosse)
Jerry Ross	Lane Community College
Patricia Roth	Southeastern Polytechnic State University
Carolyn Schauble	Colorado State University
Arjit Sengupta	Georgia State University
Bennet Setzer	Kennesaw State University
Vijay Srinivasan	JavaSoft, Sun Microsystems, Inc.
Stuart Steiner	Eastern Washington University
Katherine St. John	Lehman College, CUNY
Alexander Stoytchev	Iowa State University
Ed Timmerman	University of Maryland, University College
Shengru Tu	University of New Orleans
Paul Tymann	Rochester Institute of Technology
John J. Wegis	JavaSoft, Sun Microsystems, Inc.
Linda Wilson	Dartmouth College
David Wittenberg	Brandeis University
Wang-Chan Wong	California State University (Dominguez Hills)

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 Niezgodna, 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.

Contents

Preface		vii
Chapter 1	Introduction	1
	1.1 Computer Processing	2
	Software Categories	3
	Digital Computers	4
	Binary Numbers	7
	1.2 Hardware Components	10
	Computer Architecture	11
	Input/Output Devices	12
	Main Memory and Secondary Memory	13
	The Central Processing Unit	17
	1.3 Networks	20
	Network Connections	20
	Local-Area Networks and Wide-Area Networks	22
	The Internet	23
	The World Wide Web	24
	Uniform Resource Locators	25
	1.4 The Java Programming Language	26
	A Java Program	27
	Comments	29
	Identifiers and Reserved Words	31
	White Space	33
	1.5 Program Development	36
	Programming Language Levels	36
	Editors, Compilers, and Interpreters	38
	Development Environments	40
	Syntax and Semantics	41
	Errors	42

	1.6 Object-Oriented Programming	44
	Problem Solving	45
	Object-Oriented Software Principles	46
Chapter 2	Data and Expressions	57
	2.1 Character Strings	58
	The <code>print</code> and <code>println</code> Methods	58
	String Concatenation	60
	Escape Sequences	63
	2.2 Variables and Assignment	65
	Variables	65
	The Assignment Statement	67
	Constants	69
	2.3 Primitive Data Types	71
	Integers and Floating Points	71
	Characters	73
	Booleans	74
	2.4 Expressions	75
	Arithmetic Operators	75
	Operator Precedence	76
	Increment and Decrement Operators	80
	Assignment Operators	81
	2.5 Data Conversion	83
	Conversion Techniques	85
	2.6 Interactive Programs	87
	The Scanner Class	87
	2.7 Graphics	92
	Coordinate Systems	92
	Representing Color	94
	2.8 Applets	95
	Executing Applets Using the Web	98
	2.9 Drawing Shapes	99
	The Graphics Class	99
	Software Failure:	
	NASA Mars Climate Orbiter and Polar Lander	111

Chapter 3	Using Classes and Objects	113
3.1	Creating Objects	114
	Aliases	116
3.2	The String Class	118
3.3	Packages	122
	The import Declaration	124
3.4	The Random Class	126
3.5	The Math Class	129
3.6	Formatting Output	132
	The NumberFormat Class	132
	The DecimalFormat Class	134
	The printf Method	135
3.7	Enumerated Types	138
3.8	Wrapper Classes	141
	Autoboxing	143
3.9	Components and Containers	143
	Frames and Panels	144
3.10	Nested Panels	148
3.11	Images	151
Chapter 4	Writing Classes	159
4.1	Classes and Objects Revisited	160
4.2	Anatomy of a Class	162
	Instance Data	167
	UML Class Diagrams	167
4.3	Encapsulation	169
	Visibility Modifiers	170
	Accessors and Mutators	171
4.4	Anatomy of a Method	172
	The return Statement	174
	Parameters	175