

GLOBAL
EDITION



JavaTM Software Solutions

Foundations of Program Design

NINTH EDITION

Lewis • Loftus



Digital Resources for Students

Your new textbook provides 12-month access to digital resources that may include VideoNotes (step-by-step video tutorials on programming concepts), source code, web chapters, quizzes, and more. Refer to the preface in the textbook for a detailed list of resources.

Follow the instructions below to register for the Companion Website for John Lewis and William Loftus' *Java™ Software Solutions*, Ninth Edition, Global Edition.

1. Go to www.pearsonglobaleditions.com/Lewis
2. Select your textbook and click Companion Website.
3. Click Register and follow the on-screen instructions to create a login name and password.

**Use a coin to scratch off the coating and reveal your access code.
Do not use a sharp knife or other sharp object as it may damage the code.**

Use the login name and password you created during registration to start using the online resources that accompany your textbook.

IMPORTANT

This prepaid subscription does not include access to Pearson MyLab Programming, which is available at www.myprogramminglab.com for purchase.

This access code can only be used once. This subscription is valid for 12 months upon activation and is not transferable. If the access code has already been revealed, it may no longer be valid.

For technical support, go to <https://support.pearson.com/getsupport>

javaTM

SOFTWARE SOLUTIONS

Ninth Edition
Global Edition

FOUNDATIONS OF PROGRAM DESIGN

JOHN LEWIS

Virginia Tech

•

WILLIAM LOFTUS

Accenture



 Pearson

330 Hudson Street, NY NY 10013

Director, Portfolio Management: Engineering, Computer
Science & Global Editions: *Julian Partridge*
Specialist, Higher Ed Portfolio Management: *Matt Goldstein*
Portfolio Management Assistant: *Kristy Alaura*
Assistant Acquisitions Editor, Global Edition: *Aditee
Agarwal*
Senior Project Editor, Global Edition: *Amrita Naskar*
Managing Content Producer: *Scott Disanno*
Content Producer: *Carole Snyder*
Web Developer: *Steve Wright*
Senior Media Editor, Global Edition: *Gargi Banerjee*
Senior Manufacturing Controller, Global Edition: *Jerry
Kataria*

Rights and Permissions Manager: *Ben Ferrini*
Manufacturing Buyer, Higher Ed, Lake Side
Communications Inc (LSC): *Maura Zaldivar-Garcia*
Inventory Manager: *Ann Lam*
Product Marketing Manager: *Yvonne Vannatta*
Field Marketing Manager: *Demetrius Hall*
Marketing Assistant: *Jon Bryant*
Cover Designer: *Lumina Datamatics Ltd.*
Cover Photo Credit: *fon.tepsoda/Shutterstock*
Project Management: *Louise C. Capulli, Lakeside Editorial
Services, L.L.C.*
Full-Service Project Management: *Revathi Viswanathan,
Cenveo® Publisher Services*

Credits and acknowledgments borrowed from other sources and reproduced, with permission, appear on the Credits page at the end of the front matter of this textbook.

Pearson Education Limited
KAO Two
KAO Park
Harlow
Essex CM17 9NA
England

and Associated Companies throughout the world

Visit us on the World Wide Web at:
www.pearsonglobaleditions.com

© Pearson Education Limited 2018

The rights of John Lewis and William Loftus to be identified as the authors of this work have been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

Authorized adaptation from the United States edition, entitled Java™ Software Solutions: Foundations of Program Design, 9th edition, ISBN 9780134462028, by John Lewis and William Loftus, published by Pearson Education © 2017.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without either the prior written permission of the publisher or a license permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6–10 Kirby Street, London EC 1N 8TS.

All trademarks used herein are the property of their respective owners. The use of any trademark in this text does not vest in the author or publisher any trademark ownership rights in such trademarks, nor does the use of such trademarks imply any affiliation with or endorsement of this book by such owners.

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.

The programs and applications presented in this book have been included for their instructional value. They have been tested with care, but are not guaranteed for any particular purpose. The publisher does not offer any warranties or representations, nor does it accept any liabilities with respect to the programs or applications.

ISBN 10: 1-292-22172-0
ISBN 13: 978-1-292-22172-4

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1

Typeset by Cenveo Publisher Services

Printed and bound by Vivar in Malaysia

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

This page intentionally left blank

Preface

Welcome to the Ninth 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

The biggest change to this edition of *Java Software Solutions* is a sweeping overhaul of the Graphics Track sections of the book to fully embrace the JavaFX API. Swing is no longer actively supported by Oracle. JavaFX is now the preferred approach for developing graphics and graphical user interfaces (GUIs) in Java, and we make use of it throughout this text.

The changes include the following:

- Coverage of JavaFX graphical shapes.
- Coverage of JavaFX controls, including buttons, text fields, check boxes, radio buttons, choice boxes, color pickers, date pickers, dialog boxes, sliders, and spinners.
- Use of Java 8 method references and lambda expressions to define event handlers.
- An exploration of the JavaFX class hierarchy.
- An explanation of JavaFX properties and property binding.
- Revised end-of-chapter exercises and programming projects.
- A new appendix (Appendix G) that presents an overview of JavaFX layout panes.
- A new appendix (Appendix H) that introduces the JavaFX Scene Builder software.

There are two exciting aspects to embracing JavaFX. First, it provides a much cleaner approach to GUI development than Swing did. Equivalent programs using JavaFX are shorter and more easily understood.

Second, the JavaFX approach embraces core object-oriented principles better than Swing did. For example, all graphic shapes are represented by classes with fundamental data elements, such as a `Circle` class with a radius. Early on (Chapter 3), the shape classes provide a wealth of basic, well-designed classes, just when students need to understand what classes and objects are all about.

The use of Java 8 method references provides an easy-to-understand approach to defining event handlers. The use of the (underlying) lambda expressions is also explored as an alternative approach.

JavaFX layout panes are used and explained as needed in examples, with a full overview of layout panes provided in a new appendix. We think this works better than the way we treated Swing layout managers, as a separate topic in a chapter.

All GUI development in the book is done “by hand” in straight Java code, which is important for beginning students. The JavaFX drag-and-drop Scene Builder is discussed in a new appendix, but it is not used in the book itself.

In addition to the changes related to JavaFX, we also updated examples and discussions in various places throughout the book as needed to bring them up-to-date and improve their pedagogy.

We’re excited about the opportunities this new edition of Java Software Solutions provides for both students and instructors. As always, questions and comments are welcome.

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. The book fully embraces the JavaFX API, the preferred and fully-supported approach to Java graphics 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.pearsonglobaleditions.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, rewind, 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 with Pearson MyLab Programming

Pearson MyLab Programming helps students fully grasp the logic, semantics, and syntax of programming. Through practice exercises and immediate, personalized feedback, Pearson MyLab Programming 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, Pearson MyLab Programming consists of hundreds of small practice exercises 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.

Pearson MyLab Programming 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 Pearson MyLab Programming 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.pearsonglobaleditions.com/Lewis) for information on how to access them:

- Presentation Slides—in PowerPoint.
- Solutions to end-of-chapter Exercises.
- Solutions to end-of-chapter Programming Projects.

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.

Pearson MyLab Programming. Through practice exercises and immediate, personalized feedback, Pearson MyLab Programming 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.

Bradley Richards, at the University of Applied Sciences in Northwestern Switzerland, provided helpful advice and resources during the transition to JavaFX. Brian Fraser of Simon Fraser University also has provided some excellent

feedback that helped clarify some issues. Such interaction with computing educators is incredibly valuable.

I also want to thank Dan Joyce from Villanova University, who developed the original 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. Matt Goldstein, our editor, has amazing insight and commitment. His assistant, Kristy Alaura, is a source of consistent and helpful support. Marketing Manager Demetrius Hall makes sure that instructors understand the pedagogical advantages of the text. The cover was designed by the skilled talents of Joyce Wells. Scott Disanno and Carole Snyder led the production effort. Louise Capulli, of Lakeside Editorial Services, was the Project Manager for this edition and a huge help to the author on a daily basis. 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
Hossein Assadipour	Rutgers 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
Don Braffitt	Radford University
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
Sherif Elfayoumy	University of North 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
Daniel Joyce	Villanova University
Saroja Kanchi	Kettering University
Gregory Kapfhammer	Allegheny College
Karen Kluge	Dartmouth College
Jason Levy	University of Hawaii
Peter MacKenzie	McGill University
Jerry Marsh	Oakland 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)
Bob Roos	Allegheny College
Carolyn Rosiene	University of Hartford
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.
Ken Williams	North Carolina Agricultural and Technical University
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. Thanks also to Pete DePasquale, formerly of The College of New Jersey and now with SailThru, Inc.

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.

Global Edition Acknowledgments

The publishers would like to thank the following for their contribution to the Global Edition:

Contributor

Vincent Ramdhanie School of Business and Computer Science (SBCS)

Reviewers

Muthuraj M. Samsung
Patricia Moore Dublin City University
Subrata Sinha Dibrugarh University

This page intentionally left blank

Contents

Preface		5
Chapter 1	Introduction	29
1.1	Computer Processing	30
	Software Categories	31
	Digital Computers	33
	Binary Numbers	35
1.2	Hardware Components	38
	Computer Architecture	39
	Input/Output Devices	40
	Main Memory and Secondary Memory	41
	The Central Processing Unit	45
1.3	Networks	47
	Network Connections	48
	Local-Area Networks and Wide-Area Networks	49
	The Internet	50
	The World Wide Web	52
	Uniform Resource Locators	53
1.4	The Java Programming Language	54
	A Java Program	55
	Comments	57
	Identifiers and Reserved Words	58
	White Space	61
1.5	Program Development	63
	Programming Language Levels	64
	Editors, Compilers, and Interpreters	66
	Development Environments	68
	Syntax and Semantics	68
	Errors	69

	1.6	Object-Oriented Programming	71
		Problem Solving	72
		Object-Oriented Software Principles	73
Chapter 2		Data and Expressions	83
	2.1	Character Strings	84
		The <code>print</code> and <code>println</code> Methods	84
		String Concatenation	86
		Escape Sequences	89
	2.2	Variables and Assignment	91
		Variables	91
		The Assignment Statement	93
		Constants	95
	2.3	Primitive Data Types	97
		Integers and Floating Points	97
		Characters	99
		Booleans	100
	2.4	Expressions	101
		Arithmetic Operators	101
		Operator Precedence	102
		Increment and Decrement Operators	106
		Assignment Operators	107
	2.5	Data Conversion	109
		Conversion Techniques	111
	2.6	Interactive Programs	113
		The <code>Scanner</code> Class	113
		Software Failure:	
		NASA Mars Climate Orbiter and Polar Lander	124
Chapter 3		Using Classes and Objects	127
	3.1	Creating Objects	128
		Aliases	130
	3.2	The <code>String</code> Class	132

3.3	Packages	136
	The <code>import</code> Declaration	138
3.4	The <code>Random</code> Class	140
3.5	The <code>Math</code> Class	143
3.6	Formatting Output	146
	The <code>NumberFormat</code> Class	146
	The <code>DecimalFormat</code> Class	148
	The <code>printf</code> Method	149
3.7	Enumerated Types	152
3.8	Wrapper Classes	155
	Autoboxing	157
3.9	Introduction to JavaFX	157
3.10	Basic Shapes	161
3.11	Representing Colors	168
Chapter 4	Writing Classes	175
4.1	Classes and Objects Revisited	176
4.2	Anatomy of a Class	178
	Instance Data	183
	UML Class Diagrams	183
4.3	Encapsulation	185
	Visibility Modifiers	186
	Accessors and Mutators	187
4.4	Anatomy of a Method	188
	The <code>return</code> Statement	190
	Parameters	191
	Local Data	191
	Bank Account Example	192
4.5	Constructors Revisited	197
4.6	Arcs	198
4.7	Images	201
	Viewports	203

	4.8 Graphical User Interfaces	204
	Alternate Ways to Specify Event Handlers	207
	4.9 Text Fields	208
	Software Failure:	
	Denver Airport Baggage Handling System	217
Chapter 5	Conditionals and Loops	219
	5.1 Boolean Expressions	220
	Equality and Relational Operators	221
	Logical Operators	222
	5.2 The if Statement	225
	The <code>if-else</code> Statement	228
	Using Block Statements	231
	Nested <code>if</code> Statements	235
	5.3 Comparing Data	238
	Comparing Floats	238
	Comparing Characters	239
	Comparing Objects	240
	5.4 The while Statement	242
	Infinite Loops	246
	Nested Loops	248
	The <code>break</code> and <code>continue</code> Statements	251
	5.5 Iterators	253
	Reading Text Files	254
	5.6 The ArrayList Class	257
	5.7 Determining Event Sources	260
	5.8 Managing Fonts	262
	5.9 Check Boxes	265
	5.10 Radio Buttons	269
	Software Failure:	
	Therac-25	281

Chapter 6	More Conditionals and Loops	283
6.1	The switch Statement	284
6.2	The Conditional Operator	288
6.3	The do Statement	289
6.4	The for Statement	293
	The for-each Loop	296
	Comparing Loops	298
6.5	Using Loops and Conditionals with Graphics	299
6.6	Graphic Transformations	304
	Translation	304
	Scaling	304
	Rotation	305
	Shearing	306
	Applying Transformations on Groups	307
Chapter 7	Object-Oriented Design	317
7.1	Software Development Activities	318
7.2	Identifying Classes and Objects	319
	Assigning Responsibilities	321
7.3	Static Class Members	321
	Static Variables	322
	Static Methods	322
7.4	Class Relationships	326
	Dependency	326
	Dependencies Among Objects of the Same Class	326
	Aggregation	332
	The <code>this</code> Reference	336
7.5	Interfaces	338
	The <code>Comparable</code> Interface	343
	The <code>Iterator</code> Interface	344

	7.6 Enumerated Types Revisited	345
	7.7 Method Design	348
	Method Decomposition	349
	Method Parameters Revisited	354
	7.8 Method Overloading	359
	7.9 Testing	361
	Reviews	362
	Defect Testing	362
	7.10 GUI Design	365
	7.11 Mouse Events	366
	7.12 Key Events	371
	Software Failure:	
	2003 Northeast Blackout	380
Chapter 8	Arrays	383
	8.1 Array Elements	384
	8.2 Declaring and Using Arrays	385
	Bounds Checking	388
	Alternate Array Syntax	393
	Initializer Lists	393
	Arrays as Parameters	394
	8.3 Arrays of Objects	396
	8.4 Command-Line Arguments	406
	8.5 Variable Length Parameter Lists	408
	8.6 Two-Dimensional Arrays	412
	Multidimensional Arrays	416
	8.7 Polygons and Polylines	417
	8.8 An Array of Color Objects	420
	8.9 Choice Boxes	423
	Software Failure:	
	LA Air Traffic Control	433

Chapter 9	Inheritance	435
9.1	Creating Subclasses	436
	The <code>protected</code> Modifier	439
	The <code>super</code> Reference	442
	Multiple Inheritance	445
9.2	Overriding Methods	447
	Shadowing Variables	449
9.3	Class Hierarchies	450
	The <code>Object</code> Class	452
	Abstract Classes	453
	Interface Hierarchies	455
9.4	Visibility	455
9.5	Designing for Inheritance	458
	Restricting Inheritance	459
9.6	Inheritance in JavaFX	460
9.7	Color and Date Pickers	462
9.8	Dialog Boxes	466
	File Choosers	469
	Software Failure:	
	Ariane 5 Flight 501	477
Chapter 10	Polymorphism	479
10.1	Late Binding	480
10.2	Polymorphism via Inheritance	481
10.3	Polymorphism via Interfaces	494
10.4	Sorting	496
	Selection Sort	497
	Insertion Sort	503
	Comparing Sorts	504

10.5	Searching	505
	Linear Search	505
	Binary Search	507
	Comparing Searches	511
10.6	Designing for Polymorphism	511
10.7	Properties	513
	Change Listeners	516
10.8	Sliders	519
10.9	Spinners	521
Chapter 11	Exceptions	529
11.1	Exception Handling	530
11.2	Uncaught Exceptions	531
11.3	The try-catch Statement	532
	The finally Clause	536
11.4	Exception Propagation	537
11.5	The Exception Class Hierarchy	541
	Checked and Unchecked Exceptions	544
11.6	I/O Exceptions	545
11.7	Tool Tips and Disabling Controls	549
11.8	Scroll Panes	553
11.9	Split Panes and List Views	556
Chapter 12	Recursion	565
12.1	Recursive Thinking	566
	Infinite Recursion	566
	Recursion in Math	567
12.2	Recursive Programming	568
	Recursion vs. Iteration	571
	Direct vs. Indirect Recursion	571

12.3	Using Recursion	572
	Traversing a Maze	573
	The Towers of Hanoi	578
12.4	Tiled Images	583
12.5	Fractals	587
Chapter 13	Collections	601
13.1	Collections and Data Structures	602
	Separating Interface from Implementation	602
13.2	Dynamic Representations	603
	Dynamic Structures	603
	A Dynamically Linked List	604
	Other Dynamic List Representations	609
13.3	Linear Collections	611
	Queues	611
	Stacks	612
13.4	Non-Linear Data Structures	615
	Trees	615
	Graphs	616
13.5	The Java Collections API	618
	Generics	618
Appendix A	Glossary	625
Appendix B	Number Systems	649
Appendix C	The Unicode Character Set	657
Appendix D	Java Operators	661
Appendix E	Java Modifiers	667
Appendix F	Java Coding Guidelines	671

Appendix G	JavaFX Layout Panes	677
Appendix H	JavaFX Scene Builder	687
Appendix I	Regular Expressions	697
Appendix J	Javadoc Documentation Generator	699
Appendix K	Java Syntax	705
Appendix L	Answers to Self-Review Questions	719
Index		773

VideoNote

Overview of program elements.	56
Comparison of Java IDEs.	68
Examples of various error types.	70
Developing a solution for PP 1.2.	81
Example using strings and escape sequences.	89
Review of primitive data and expressions.	102
Example using the <code>Scanner</code> class.	117
Developing a solution of PP 2.10.	122
Creating objects.	129
Example using the <code>Random</code> and <code>Math</code> classes.	143
Developing a solution of PP 3.6.	173
Dissecting the <code>Die</code> class.	180
Discussion of the <code>Account</code> class.	194
Developing a solution of PP 4.2.	214
Examples using conditionals.	233
Examples using <code>while</code> loops.	245
Developing a solution of PP 5.4.	278
Examples using <code>for</code> loops.	294
Developing a solution of PP 6.2.	313
Exploring the <code>static</code> modifier.	321
Examples of method overloading.	360
Developing a solution of PP 7.1.	377
Overview of arrays.	387
Discussion of the <code>LetterCount</code> example.	392
Developing a solution of PP 8.5.	431
Overview of inheritance.	441
Example using a class hierarchy.	453
Exploring the <code>Firm</code> program.	482

Sorting Comparable objects.	502
Developing a solution of PP 10.1.	526
Proper exception handling.	537
Developing a solution of PP 11.1.	562
Tracing the <code>MazeSearch</code> program.	576
Exploring the Towers of Hanoi.	579
Developing a solution of PP 12.1.	597
Example using a linked list.	604
Implementing a queue.	612
Developing a solution of PP 13.3.	622

Credits

Cover: fon.tepsoda/Shutterstock

Figure 2.1: NASA EOS Earth Observing System

Figure 4.2: Susan Van Etten/PhotoEdit

Figure 5.1: David Joel/The Image Bank/Getty Images

Figures 4.1a, 4.1b, 6.1a, 6.1b, 6.2, 11.3, 11.8, 11.9, 11.10, 12.5, APG.1a,
APG.1b: Pixabay

Figures 7.1a and 7.1b: Anarres/Openclipart

Figures 7.2a and 7.2b: National Oceanic and Atmospheric Administration

Figure 8.1: Matthew McVay/The Image Bank/Getty Images

Figure 9.1: Mario Fourmy/Redux Pictures

Figure H.1: NASA

BREAKTHROUGH

To improving results



get with the programming

Through the power of practice and immediate personalized feedback, Pearson MyLab Programming improves your performance.

Pearson MyLabTM Programming

Learn more at www.myprogramminglab.com

Introduction



CHAPTER OBJECTIVES

- Describe the relationship between hardware and software.
- Define various types of software and how they are used.
- Identify the core hardware components of a computer and explain their roles.
- Explain how the hardware components interact to execute programs and manage data.
- Describe how computers are connected into networks to share information.
- Introduce the Java programming language.
- Describe the steps involved in program compilation and execution.
- Present an overview of object-oriented principles.

This book is about writing well-designed software. To understand software, we must first have a fundamental understanding of its role in a computer system. Hardware and software cooperate in a computer system to accomplish complex tasks. The purpose of various hardware components and the way those components are connected into networks are important prerequisites to the study of software development. This chapter first discusses basic computer processing and then begins our exploration of software development by introducing the Java programming language and the principles of object-oriented programming.

1.1 Computer Processing

All computer systems, whether it's a desktop, laptop, tablet, smartphone, gaming console, or a special-purpose device such as a car's navigation system, share certain characteristics. The details vary, but they all process data in similar ways. While the majority of this book deals with the development of software, we'll begin with an overview of computer processing to set the context. It's important to establish some fundamental terminology and see how key pieces of a computer system interact.

A computer system is made up of hardware and software. The *hardware* components of a computer system are the physical, tangible pieces that support the computing effort. They include chips, boxes, wires, keyboards, speakers, disks, memory cards, universal serial bus (USB) flash drives (also called jump drives), cables, plugs, printers, mice, monitors, routers, and so on. If you can physically touch it and it can be considered part of a computer system, then it is computer hardware.

KEY CONCEPT

A computer system consists of hardware and software that work in concert to help us solve problems.

The hardware components of a computer are essentially useless without instructions to tell them what to do. A *program* is a series of instructions that the hardware executes one after another. *Software* consists of programs and the data that programs use. Software is the intangible counterpart to the physical hardware components.

Together they form a tool that we can use to help solve problems.

The key hardware components in a computer system are

- central processing unit (CPU)
- input/output (I/O) devices
- main memory
- secondary memory devices

Each of these hardware components is described in detail in the next section. For now, let's simply examine their basic roles. The *central processing unit (CPU)* is a device that executes the individual commands of a program. *Input/output (I/O) devices*, such as the keyboard, mouse, trackpad, and monitor, allow a human being to interact with the computer.

Programs and data are held in storage devices called memory, which fall into two categories: main memory and secondary memory. *Main memory* is the storage device that holds the software while it is being processed by the CPU. *Secondary memory* devices store software in a relatively permanent manner. The most important secondary memory device of a typical computer system is the hard disk that resides inside the main computer box. A USB flash drive is also an important secondary memory device. A typical USB flash drive cannot store nearly as much information as a hard disk. USB flash drives have the advantage of portability; they can be removed temporarily or moved from computer to computer as needed.