PREFACE

Dear Reader,

what is new?

Many of you have provided feedback on earlier editions of this book, and your comments and suggestions have greatly improved the book. This edition has been substantially enhanced in presentation, organization, examples, exercises, and supplements. We have:

- Reorganized sections and chapters to present the subjects in a more logical order
- Included many new interesting examples and exercises to stimulate interests
- Updated to Java 7
- Created animations for algorithms and data structures to visually demonstrate the concepts
- Redesigned the support Website to make it easier to navigate

problem-driven

This book teaches programming in a problem-driven way that focuses on problem solving rather than syntax. We make introductory programming interesting by using thought-provoking problems in a broad context. The central thread of early chapters is on problem solving. Appropriate syntax and library are introduced to enable readers to write programs for solving the problems. To support the teaching of programming in a problem-driven way, the book provides a wide variety of problems at various levels of difficulty to motivate students. To appeal to students in all majors, the problems cover many application areas, including math, science, business, financial, gaming, animation, and multimedia.

fundamentals-first

The book focuses on fundamentals first by introducing basic programming concepts and techniques before designing custom classes. The fundamental concepts and techniques of loops, methods, and arrays are the foundation for programming. Building this strong foundation prepares students to learn object-oriented programming and advanced Java programming.

comprehensive version

This *comprehensive version* covers fundamentals of programming, object-oriented programming, GUI programming, algorithms and data structures, concurrency, networking, internationalization, advanced GUI, database, and Web programming. It is designed to prepare students to become proficient Java programmers. A *brief version* (*Introduction to Java Programming*, Brief Version, Ninth Edition) is available for a first course on programming, commonly known as CS1. The brief version contains the first 20 chapters of the comprehensive version.

brief version

examples and exercises

The best way to teach programming is *by example*, and the only way to learn programming is *by doing*. Basic concepts are explained by example, and a large number of exercises with various levels of difficulty are provided for students to practice. For our programming courses, we assign programming exercises after each lecture.

Our goal is to produce a text that teaches problem solving and programming in a broad context using a wide variety of interesting examples. If you have any comments on and suggestions for improving the book, please email me.

Sincerely,

Y. Daniel Liang
y.daniel.liang@gmail.com
www.cs.armstrong.edu/liang
www.pearsonhighered.com/liang

What's New in This Edition?

This edition substantially improves *Introduction to Java Programming*, Eighth Edition. The major improvements are as follows:

- This edition is completely revised in every detail to enhance clarity, presentation, content, examples, and exercises. — complete revision
- New examples and exercises are provided to motivate and stimulate student interest in programming. — new problems
- Each section starts with a Key Point that highlights the important concepts covered in the section. — key point
- Check Points provide review questions to help students track their progress and evaluate their learning after a major concept or example is covered. — check point
- Each chapter provides test questions online. They are grouped by sections for students to do self-test. The questions are graded online. — test questions
- New VideoNotes provide short video tutorials designed to reinforce code. — VideoNotes
- The Java GUI API is an excellent example of how the object-oriented principle is applied. Students learn better with concrete and visual examples. So basic GUI/Graphics is moved before introducing abstract classes and interfaces. You can however still choose to cover abstract classes and interfaces before GUI or skip GUI. — basic GUI and graphics early
- The numeric wrapper classes, `BigInteger`, and `BigDecimal` are now introduced in Chapter 10 to enable students to write code using these classes early. — numeric classes covered early
- Exception handling is covered before abstract classes and interfaces so that students can build robust programs early. The instructor can still choose to cover exception handling later. Text I/O is now combined with exception handling to form a new chapter. — exception handling earlier
- Simple use of generics is introduced along with `ArrayList` in Chapter 11 and with `Comparable` in Chapter 15 while the complex detail on generics is still kept in Chapter 21. — simple generics early
- Chapter 22 is split into two chapters (Chapter 22 and Chapter 23) to make room for incorporating three new case studies to demonstrate effective use of data structures. — splitting Chapter 22
- Chapter 24 is expanded to introduce algorithmic techniques: dynamic programming, divide-and-conquer, backtracking, and greedy algorithm with new examples to design efficient algorithms. — developing efficient algorithms
- Visual animations are created to show how data structures and algorithms work. — data structures and algorithm animation
- A common problem with a data structures course is lack of good examples and exercises. This edition added many new interesting examples and exercises. — new data structures materials
- Parallel programming techniques are introduced in Chapter 32, Multithreading and Parallel Programming. — parallel programming
- Chapter 44 is completely new to introduce the latest standard on JSF. — new JSF chapter
- Chapter 50 is completely new to introduce testing using JUnit. — new JUnit chapter

Please visit www.cs.armstrong.edu/liang/intro9e/newfeatures.html for a complete list of new features as well as correlations to the previous edition.

Pedagogical Features

The book uses the following elements to help students get the most from the material:

- The **Objectives** at the beginning of each chapter list what students should learn from the chapter. This will help them determine whether they have met the objectives after completing the chapter.
- The **Introduction** opens the discussion with representative problems to give the reader an overview of what to expect from the chapter.
- **Key Points** highlight the important concepts covered in each section.
- **Check Points** provide review questions to help students track their progress as they read through the chapter and evaluate their learning.
- **Problems and Case Studies**, carefully chosen and presented in an easy-to-follow style, teach problem solving and programming concepts. The book uses many small, simple, and stimulating examples to demonstrate important ideas.
- The **Chapter Summary** reviews the important subjects that students should understand and remember. It helps them reinforce the key concepts they have learned in the chapter.
- **Test Questions** are accessible online, grouped by sections, for students to do self-test on programming concepts and techniques.
- **Programming Exercises** are grouped by sections to provide students with opportunities to apply the new skills they have learned on their own. The level of difficulty is rated as easy (no asterisk), moderate (*), hard (**), or challenging (***). The trick of learning programming is practice, practice, and practice. To that end, the book provides a great many exercises.
- **Notes**, **Tips**, **Cautions**, and **Design Guides** are inserted throughout the text to offer valuable advice and insight on important aspects of program development.

Note
Provides additional information on the subject and reinforces important concepts.

Tip
Teaches good programming style and practice.

Caution
Helps students steer away from the pitfalls of programming errors.

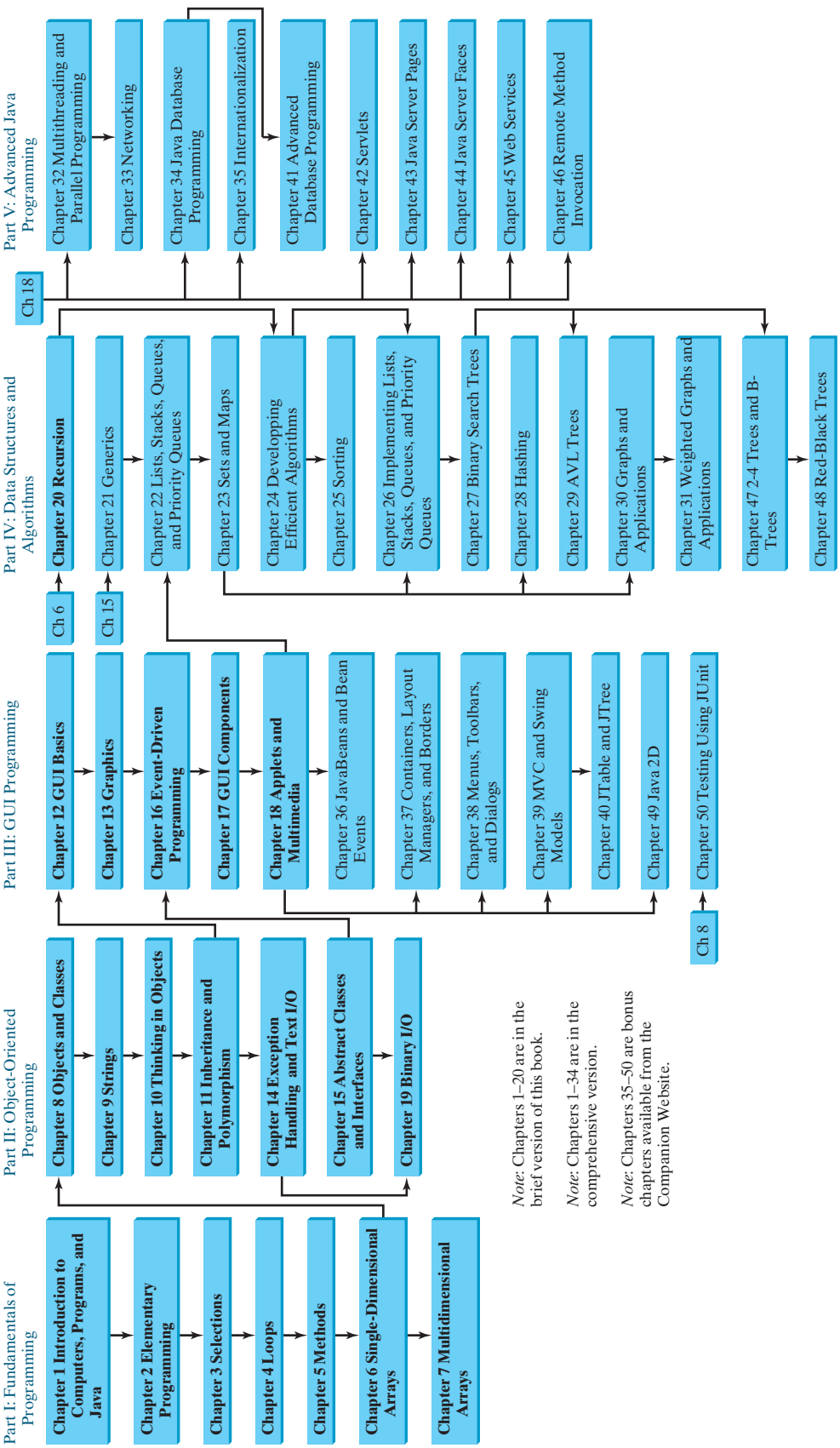
Design Guide
Provides guidelines for designing programs.

Flexible Chapter Orderings

The book is designed to provide flexible chapter orderings to enable GUI, exception handling, recursion, generics, and the Java Collections Framework to be covered earlier or later. The diagram on the next page shows the chapter dependencies.

Organization of the Book

The chapters can be grouped into five parts that, taken together, form a comprehensive introduction to Java programming, data structures and algorithms, and database and Web programming. Because knowledge is cumulative, the early chapters provide the conceptual basis



for understanding programming and guide students through simple examples and exercises; subsequent chapters progressively present Java programming in detail, culminating with the development of comprehensive Java applications. The appendixes contain a mixed bag of topics, including an introduction to number systems and bitwise operations.

Part I: Fundamentals of Programming (Chapters 1–7)

The first part of the book is a stepping stone, preparing you to embark on the journey of learning Java. You will begin to learn about Java (Chapter 1) and fundamental programming techniques with primitive data types, variables, constants, assignments, expressions, and operators (Chapter 2), control statements (Chapters 3–4), methods (Chapter 5), and arrays (Chapters 6–7). After Chapter 6, you can jump to Chapter 20 to learn how to write recursive methods for solving inherently recursive problems.

Part II: Object-Oriented Programming (Chapters 8–11, 14–15, and 19)

This part introduces object-oriented programming. Java is an object-oriented programming language that uses abstraction, encapsulation, inheritance, and polymorphism to provide great flexibility, modularity, and reusability in developing software. You will learn programming with objects and classes (Chapters 8–10), class inheritance (Chapter 11), polymorphism (Chapter 11), exception handling and text I/O (Chapter 14), abstract classes (Chapter 15), and interfaces (Chapter 15). Processing strings is introduced in Chapter 9, and binary I/O is discussed in Chapter 19.

Part III: GUI Programming (Chapters 12–13, 16–18, and Bonus Chapters 36–40 and 49)

This part introduces elementary Java GUI programming in Chapters 12–13 and 16–18 and advanced Java GUI programming in Chapters 36–40 and 49. Major topics include GUI basics (Chapter 12), drawing shapes (Chapter 13), event-driven programming (Chapter 16), using GUI components (Chapter 17), and writing applets (Chapter 18). You will learn the architecture of Java GUI programming and use the GUI components to develop applications and applets from these elementary GUI chapters. The advanced GUI chapters discuss Java GUI programming in more depth and breadth. You will delve into JavaBeans and learn how to develop custom events and source components in Chapter 36, review and explore new containers, layout managers, and borders in Chapter 37, learn how to create GUI with menus, popup menus, toolbars, dialogs, and internal frames in Chapter 38, develop components using the MVC approach and explore the advanced Swing components `JSpinner`, `JList`, and `JComboBox` in Chapter 39, and `JTable` and `JTree` in Chapter 40. Chapter 49 introduces Java 2D.

Part IV: Data Structures and Algorithms (Chapters 20–31 and Bonus Chapters 47–48)

This part covers the main subjects in a typical data structures course. Chapter 20 introduces recursion to write methods for solving inherently recursive problems. Chapter 21 presents how generics can improve software reliability. Chapters 22 and 23 introduce the Java Collection Framework, which defines a set of useful API for data structures. Chapter 24 discusses measuring algorithm efficiency in order to choose an appropriate algorithm for applications. Chapter 25 describes classic sorting algorithms. You will learn how to implement several classic data structures lists, queues, and priority queues in Chapter 26. Chapters 27 and 29 introduce binary search trees and AVL trees. Chapter 28 presents hashing and implementing maps and sets using hashing. Chapters 30 and 31 introduce graph applications. The 2-4 trees, B-trees, and red-black trees are covered in Chapters 47–48.

Part V: Advanced Java Programming (Chapters 32–33 and Bonus Chapters 35, 41–46, and 50)

This part of the book is devoted to advanced Java programming. Chapter 32 treats the use of multithreading to make programs more responsive and interactive and introduces parallel programming. Chapter 33 discusses how to write programs that talk with each other

over the Internet. Chapter 34 introduces the use of Java to develop database projects, and Chapter 35 covers the use of internationalization support to develop projects for international audiences. Chapter 41 delves into advanced Java database programming. Bonus Chapters 42, 43 and 44 introduce how to use Java servlets, JavaServer Pages, and JavaServer Faces to generate dynamic content from Web servers. Chapter 45 discusses Web services, and Chapter 46 introduces remote method invocation. Chapter 50 introduces testing Java programs using JUnit.

Appendixes

This part of the book covers a mixed bag of topics. Appendix A lists Java keywords. Appendix B gives tables of ASCII characters and their associated codes in decimal and in hex. Appendix C shows the operator precedence. Appendix D summarizes Java modifiers and their usage. Appendix E discusses special floating-point values. Appendix F introduces number systems and conversions among binary, decimal, and hex numbers. Finally, Appendix G introduces bitwise operations.

Java Development Tools

You can use a text editor, such as the Windows Notepad or WordPad, to create Java programs and to compile and run the programs from the command window. You can also use a Java development tool, such as TextPad, NetBeans, or Eclipse. These tools support an integrated development environment (IDE) for developing Java programs quickly. Editing, compiling, building, executing, and debugging programs are integrated in one graphical user interface. Using these tools effectively can greatly increase your programming productivity. TextPad is a primitive IDE tool. NetBeans and Eclipse are more sophisticated, but they are easy to use if you follow the tutorials. Tutorials on TextPad, NetBeans, and Eclipse can be found in the supplements on the Companion Website www.cs.armstrong.edu/liang/intro9e.

IDE tutorials

Online Practice and Assessment with MyProgrammingLab

MyProgrammingLab

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, a MyProgrammingLab course 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 inputted 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.

VideoNotes

VideoNote

We are excited about the new VideoNotes feature that is found in this new edition. These videos provide additional help by presenting examples of key topics and showing how to solve problems completely, from design through coding. VideoNotes are free to first time users and can be accessed by redeeming the access code in the front of this book at www.pearsonhighered.com/liang.

LiveLab

This book is accompanied by a complementary Web-based course assessment and management system for instructors. The system has four main components:

- The **Automatic Grading System** can automatically grade programs.
- The **Quiz Creation/Submission/Grading System** enables instructors to create and modify quizzes that students can take and be graded upon automatically.
- The **Peer Evaluation System** enables peer evaluations.
- **Tracking grades, attendance, etc.,** lets students track their grades, and enables instructors to view the grades of all students and to track students' attendance.

The main features of the Automatic Grading System include:

- Students can run and submit exercises. (The system checks whether their program runs correctly—students can continue to run and resubmit the program before the due date.)
- Instructors can review submissions, run programs with instructor test cases, correct them, provide feedback to students, and check plagiarism.
- Instructors can create/modify their own exercises, create public and secret test cases, assign exercises, and set due dates for the whole class or for individuals.
- Instructors can assign all the exercises in the text to students. Additionally, LiveLab provides extra exercises that are not printed in the text.
- Instructors can sort and filter all exercises and check grades (by time frame, student, and/or exercise).
- Instructors can delete students from the system.
- Students and instructors can track grades on exercises.

The main features of the Quiz System are:

- Instructors can create/modify quizzes from the test bank or a text file or create completely new tests online.
- Instructors can assign the quizzes to students and set a due date and test time limit for the whole class or for individuals.
- Students and instructors can review submitted quizzes.
- Instructors can analyze quizzes and identify students' weaknesses.
- Students and instructors can track grades on quizzes.

The main features of the Peer Evaluation System include:

- Instructors can assign peer evaluation for programming exercises.
- Instructors can view peer evaluation reports.

Student Resource Website

The Student Resource Website (www.cs.armstrong.edu/liang/intro9e) contains the following resources:

- Access to VideoNotes (www.pearsonhighered.com/liang).
- Answers to check point questions

- Solutions to even-numbered programming exercises
- Source code for the examples in the book
- Interactive self-testing (organized by sections for each chapter)
- Data structures and algorithm animations
- Errata

Instructor Resource Website

The Instructor Resource Website, accessible from www.cs.armstrong.edu/liang/intro9e, contains the following resources:

- Microsoft PowerPoint slides with interactive buttons to view full-color, syntax-highlighted source code and to run programs without leaving the slides.
- Solutions to all programming exercises. Students will have access to the solutions of even-numbered programming exercises.
- Web-based quiz generator. (Instructors can choose chapters to generate quizzes from a large database of more than two thousand questions.)
- Sample exams. Most exams have four parts:
  - Multiple-choice questions or short-answer questions
  - Correct programming errors
  - Trace programs
  - Write programs
- Projects. In general, each project gives a description and asks students to analyze, design, and implement the project.

Some readers have requested the materials from the Instructor Resource Website. Please understand that these are for instructors only. Such requests will not be answered.

Algorithm Animations

We have provided numerous animations for algorithms. These are valuable pedagogical tools to demonstrate how algorithms work. Algorithm animations can be accessed from the Companion Website.

Acknowledgments

I would like to thank Armstrong Atlantic State University for enabling me to teach what I write and for supporting me in writing what I teach. Teaching is the source of inspiration for continuing to improve the book. I am grateful to the instructors and students who have offered comments, suggestions, bug reports, and praise.

This book has been greatly enhanced thanks to outstanding reviews for this and previous editions. The reviewers are: Elizabeth Adams (James Madison University), Syed Ahmed (North Georgia College and State University), Omar Aldawud (Illinois Institute of Technology), Yang Ang (University of Wollongong, Australia), Kevin Bierre (Rochester Institute of Technology), David Champion (DeVry Institute), James Chegwidden (Tarrant County College), Anup Dargar (University of North Dakota), Charles Dierbach (Towson University), Frank Ducrest (University of Louisiana at Lafayette), Erica Eddy (University of Wisconsin at Parkside), Deena

Engel (New York University), Henry A. Etlinger (Rochester Institute of Technology), James Ten Eyck (Marist College), Myers Foreman (Lamar University), Olac Fuentes (University of Texas at El Paso), Edward F. Gehringer (North Carolina State University), Harold Grossman (Clemson University), Barbara Guillot (Louisiana State University), Stuart Hansen (University of Wisconsin, Parkside), Dan Harvey (Southern Oregon University), Ron Hofman (Red River College, Canada), Stephen Hughes (Roanoke College), Vladan Jovanovic (Georgia Southern University), Edwin Kay (Lehigh University), Larry King (University of Texas at Dallas), Nana Kofi (Langara College, Canada), George Koutsogiannakis (Illinois Institute of Technology), Roger Kraft (Purdue University at Calumet), Norman Krumpe (Miami University), Hong Lin (DeVry Institute), Dan Lipsa (Armstrong Atlantic State University), James Madison (Rensselaer Polytechnic Institute), Frank Malinowski (Darton College), Tim Margush (University of Akron), Debbie Masada (Sun Microsystems), Blayne Mayfield (Oklahoma State University), John McGrath (J.P. McGrath Consulting), Hugh McGuire (Grand Valley State), Shyamal Mitra (University of Texas at Austin), Michel Mitri (James Madison University), Kenrick Mock (University of Alaska Anchorage), Frank Murgolo (California State University, Long Beach), Jun Ni (University of Iowa), Benjamin Nystuen (University of Colorado at Colorado Springs), Maureen Opkins (CA State University, Long Beach), Gavin Osborne (University of Saskatchewan), Kevin Parker (Idaho State University), Dale Parson (Kutztown University), Mark Pendergast (Florida Gulf Coast University), Richard Povinelli (Marquette University), Roger Priebe (University of Texas at Austin), Mary Ann Pumphrey (De Anza Junior College), Pat Roth (Southern Polytechnic State University), Amr Sabry (Indiana University), Carolyn Schauble (Colorado State University), David Scuse (University of Manitoba), Ashraf Shirani (San Jose State University), Daniel Spiegel (Kutztown University), Joslyn A. Smith (Florida Atlantic University) , Lixin Tao (Pace University), Ronald F. Taylor (Wright State University), Russ Tront (Simon Fraser University), Deborah Trytten (University of Oklahoma), Kent Vidrine (George Washington University), and Bahram Zartoshty (California State University at Northridge).

It is a great pleasure, honor, and privilege to work with Pearson. I would like to thank Tracy Dunkelberger and her colleagues Marcia Horton, Michael Hirsch, Matt Goldstein, Carole Snyder, Tim Huddleston, Yez Alayan, Jeff Holcomb, Kayla Smith-Tarbox, Gillian Hall, Rebecca Greenberg, and their colleagues for organizing, producing, and promoting this project.

As always, I am indebted to my wife, Samantha, for her love, support, and encouragement.