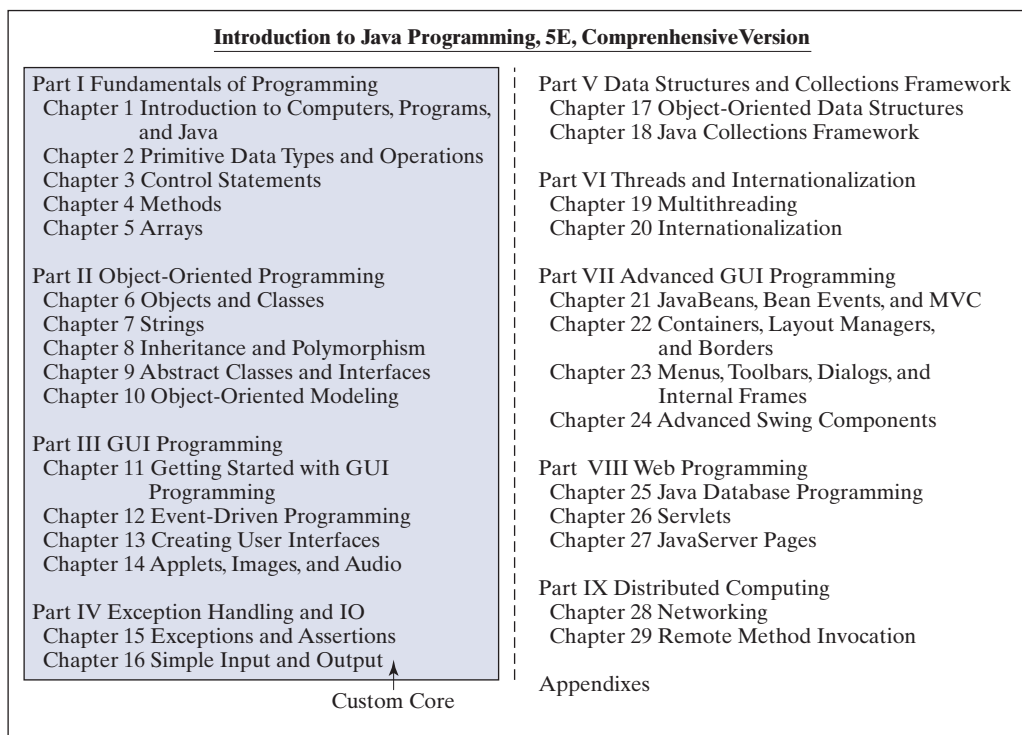


# PREFACE

In the past seven years, five editions of *Introduction to Java Programming* have been published. Each new edition substantially improved the previous edition in clarity, content, presentation, examples, and exercises, thanks to comments and suggestions by instructors and students. The Fifth Edition is a gigantic leap forward. I invite you to take a close look and be the judge. I am constantly improving the book. Please continue to send me your comments and suggestions to help further improve it.

## Custom Versions

The book is published in a comprehensive version of twenty-nine chapters and can also be printed in custom versions with substantial savings for students. The first sixteen chapters form the custom core. You can customize the book by adding new chapters to the custom core. The following diagram summarizes the materials in the comprehensive version.



Please contact your Prentice Hall sales rep to order custom versions.

## Teaching Strategies

There are three popular strategies in teaching Java. The first, known as *GUI-first*, is to mix Java applets and GUI programming with object-oriented programming concepts. The second, known

as *object-first*, is to introduce object-oriented programming from the start. The third strategy, known as *fundamentals-first*, is a step-by-step approach, first laying a sound foundation on programming concepts, control statements, methods, and arrays, then introducing object-oriented programming, and then moving on to graphical user interface (GUI), applets, and finally to exception handling, simple I/O, and other advanced subjects.

The GUI-first strategy, starting with GUI and applets, seems attractive, but requires substantial knowledge of object-oriented programming and a good understanding of the Java event-handling model; thus, students may never fully understand what they are doing.

The object-first strategy is based on the notion that objects should be introduced first because Java is an object-oriented programming language. This notion, however, overlooks the importance of the fundamental techniques required for writing programs in any programming language. Furthermore, this approach inevitably mixes static and instance variables and methods before students can fully understand classes and objects and use them to develop useful programs. Students are overwhelmed by having to master object-oriented programming and basic rules of programming simultaneously in the early stage. This is a common source of frustration for first-year students learning object-oriented programming.

From my own experience, confirmed by the experiences of many colleagues, I have found that learning basic logic and fundamental programming techniques like loops is a struggle for most first-year students. *Students who cannot write code in procedural programming are not able to learn object-oriented programming.* A good introduction on primitive data types, control statements, methods, and arrays prepares students to learn object-oriented programming. Therefore, this text adopts the fundamentals-first strategy, proceeding at a steady pace through all the necessary and important basic concepts, then moving to object-oriented programming, and then to the use of the object-oriented approach to build interesting GUI applications and applets with exception handling, simple I/O, and advanced features. The fundamentals-first approach can reinforce object-oriented programming by first presenting the procedural solutions and demonstrating how they can be improved using the object-oriented approach. Students can learn when and how to apply OOP effectively.

This book is not simply about how to program, for it teaches, as well, how to solve problems using programs. Applying the concept of abstraction in the design and implementation of software projects is the key to developing software. The overriding objective of the book, therefore, is to teach students to use many levels of abstraction in solving problems and to see problems in small and in large. *The examples and exercises throughout the book foster the concept of developing reusable components and using them to create practical projects.*

## Learning Strategies

A programming course is quite different from other courses. In a programming course, you learn from examples, from practice, and from mistakes. You need to devote a lot of time to writing programs, testing them, and fixing errors.

For first-time programmers, learning Java is like learning any high-level programming language. The fundamental point in learning programming is to develop the critical skills of formulating programmatic solutions for real problems and translating them into programs using selection statements, loops, and methods.

Once you acquire the basic skills of writing programs using loops, methods, and arrays, you can begin to learn object-oriented programming. You will learn how to develop object-oriented software using class encapsulation and class inheritance.

Once you understand the concept of object-oriented programming, learning Java becomes a matter of learning the Java API. The Java API establishes a framework for programmers to develop applications using Java. You have to use these classes and interfaces in the API and follow their conventions and rules to create applications. The best way to learn the Java API is to imitate examples and do exercises. The following diagram highlights the API covered in the book.

GUI-first

object-first

fundamentals-first

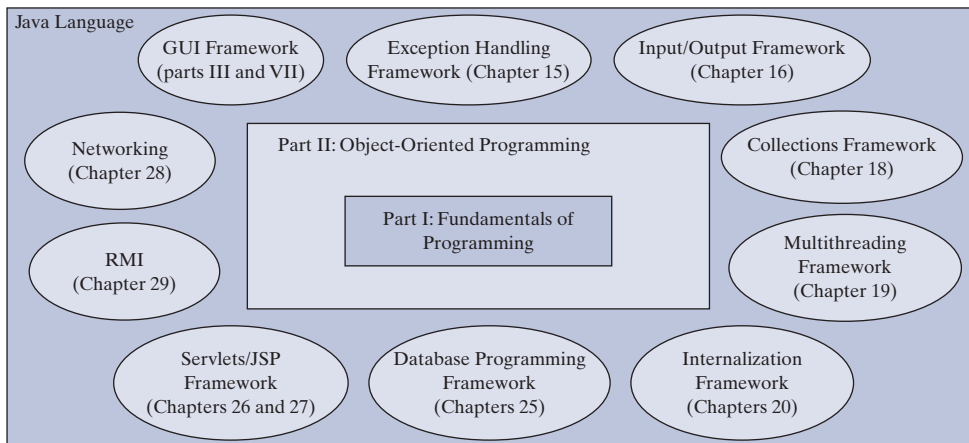
problem solving

practice

programmatic solution

object-oriented programming

Java API



## Pedagogical Features

teaching by example  
learning by doing

The philosophy of the Liang Java Series is *teaching by example and learning by doing*. Basic features are explained by example so that you can learn by doing. The book uses the following elements to get the most from the material:

- ◆ **Objectives** list what students should have learned from the chapter. This will help them to determine whether they have met the objectives after completing the chapter.
- ◆ **Introduction** opens the discussion with a brief overview of what to expect from the chapter.
- ◆ **Examples**, carefully chosen and presented in an easy-to-follow style, teach programming concepts. Each example has a problem statement, solution steps, complete source code, sample run, and review.
- ◆ **Chapter Summary** reviews the important subjects that students should understand and remember. It helps them to reinforce the key concepts they have learned in the chapter.
- ◆ **Review Questions** are grouped by sections to help students track their progress and evaluate their learning.
- ◆ **Programming Exercises** are grouped by sections to provide students with opportunities to apply the skills on their own. The level of difficulty is rated easy (no asterisk), moderate (\*), hard (\*\*), or challenging (\*\*\*). The trick of learning programming is practice, practice, and practice. To that end, the book provides a large number of exercises.
- ◆ **Interactive Self-Test** lets students test their knowledge interactively online. The Self-Test is accessible from the Companion Web site. It provides more than nine hundred multiple-choice and true/false questions organized by sections in each chapter.
- ◆ **Notes, Tips, and Cautions** 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

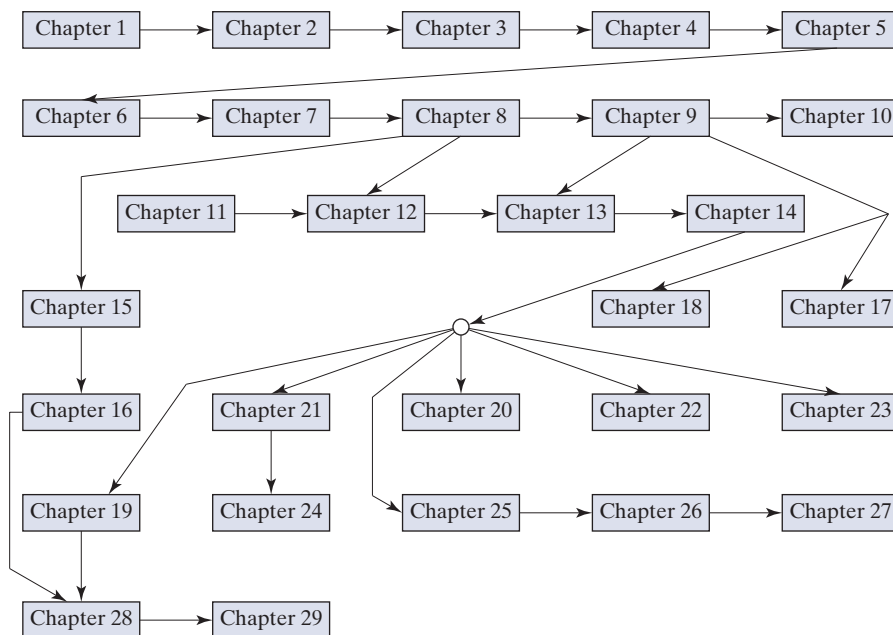
Teaches good programming style and practice.

**CAUTION**

Helps students steer away from the pitfalls of programming errors.

## Flexible Chapter Orderings

The book provides flexible chapter orderings to enable GUI, IO, or Collections to be covered earlier. Many of the chapters after Chapter 14 can be covered earlier. The following diagram shows the chapter dependencies.

**NOTE**

Some of the optional examples or exercises in a later chapter may be dependent on earlier chapters. In such cases the examples or exercises can be omitted. For example, Chapter 25 has an example that uses `JTable` from Chapter 24. If you have not covered `JTable`, this example can be skipped.

## What's New in This Edition?

This edition improves upon *Introduction to Java Programming Fourth Edition*. The major changes are as follows:

- ◆ The book is completely revised in every detail to improve clarity, content, presentation, examples, and exercises.
- ◆ The book provides many new illustrations and uses short examples to demonstrate concepts and techniques. Longer examples are presented in case studies with overall discussions and thorough line-by-line explanations.

- ◆ Part I, “Fundamentals of Programming,” focuses on problem-solving and basic programming techniques with many new illustrations and practical examples. This part uses `JOptionPane` input dialog to receive input, but console input using the `MyInput` class and the JDK 1.5 `Scanner` class are also introduced to provide alternative ways for input.
- ◆ Part II, “Object-Oriented Programming,” is expanded into five chapters to give a comprehensive introduction on OOP and how to use it to design programs. New organization improves the presentation of object-oriented programming and enables GUI programming to be covered earlier.
- ◆ Part III, “GUI Programming,” is expanded into four chapters to introduce GUI programming, event-driven programming, creating user interfaces, and applets. Advanced GUI features are now covered in Part VII, “Advanced GUI Programming.”
- ◆ Chapter 16, “Simple Input and Output,” is completely overhauled. It first introduces the `File` class, then text I/O, binary I/O, object I/O, and random access files. Short examples are used to demonstrate concepts and techniques. Three cases studies on using various I/O classes are presented in this chapter.
- ◆ The comprehensive version gives complete coverage on the Java collections framework, threads, JavaBeans, advanced GUI components, JDBC, Servlets, JSP, networking, and RMI.
- ◆ Purely mathematical examples, such as computing deviations and matrix multiplications, have been replaced by practical examples, such as computing loan payments, taxes, and printing payroll statements.
- ◆ The number of exercises is almost doubled to cover a variety of problems with simple or complex solutions. The level of difficulty is rated easy (no asterisk), moderate (\*), hard (\*\*), or challenging (\*\*\*) .
- ◆ The book is updated to JDK 1.5.

## How are the New Features in JDK 1.5 Treated?

There are already more features in Java than an introductory course can cover. This edition does not aim to cover all the new features in JDK 1.5. Nevertheless, some of the useful features of JDK 1.5 are appropriately introduced to beginners. Specifically,

- ◆ Formatted output (`System.out.printf`) is covered in Chapter 2.
- ◆ The enhanced *for* loop is covered in Chapters 5 and 18.
- ◆ The `java.util.Scanner` class is covered in Chapter 2 for console input, and in Chapter 7 to complement the `StringTokenizer` class.
- ◆ Boxing and unboxing of primitives is covered in Chapter 9.
- ◆ Static import is covered in Chapter 11.
- ◆ Generic types are covered in Chapter 18.

To facilitate the use of this book in courses based on JDK 1.4 and to enable instructors to choose JDK 1.5 topics freely, all the sections on JDK 1.5 are marked *JDK 1.5 Features* and can be skipped.

## Java Development Tools

You can use a text editor, such as the Windows Notepad, to create Java programs, and compile and run the programs from the command window. You can also use a Java development tool, such as TextPad, JBuilder, NetBeans, or Eclipse. These tools support an integrated development environment (IDE) for rapidly developing Java programs. Editing, compiling, building, and executing programs are integrated in one graphical user interface. Using these tools effectively will greatly increase your programming productivity. TextPad is a primitive IDE tool. JBuilder, NetBeans, and Eclipse are more sophisticated. It may take a while to become familiar with a tool, but the time you invest will pay off in the long run. Tutorials on TextPad, JBuilder, NetBeans, and Eclipse are in the supplements on the Companion Web site.

## Companion Web Site

The companion Web site accessible from [www.prenhall.com/liang](http://www.prenhall.com/liang) contain the following resources:

- ◆ Interactive Self-Test
- ◆ Supplements
- ◆ Download links for JDK 1.5, JBuilder, NetBeans, Eclipse, TextPad, JCreator LE, JEdit, JGrasp, BlueJ, WinZip, MySQL, and Apache Tomcat.
- ◆ Answers to review questions
- ◆ Solutions to even-numbered programming exercises
- ◆ Source code for the examples in the book

## Instructor Resource Web Site

The Instructor Resource Web site accessible from [www.prenhall.com/liang](http://www.prenhall.com/liang) 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.
- ◆ Sample exams. In general, each exam has four parts:
  1. Multiple-choice questions or short-answer questions (most of these are different from the ones in the Self-Test on the Companion Web site)
  2. Correct programming errors
  3. Trace programs
  4. Write programs
- ◆ Solutions to all the exercises. Students will have access to the solutions of even-numbered exercises from the Companion Web site.
- ◆ Quiz generator developed using Java.

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

## Supplements

The text covers the core subjects. The supplements extend the text to introduce additional topics that might be of interest to readers. The following supplements are available from the Companion Web site. New supplements may be posted as needed.

- A. Compiling and Running Java from the Command Window
- B. Compiling and Running Java from TextPad
- C. Java Coding Style Guidelines
- D. HTML Tutorial
- E. Glossary
- F. SQL statements for creating and initializing tables for Chapters 25, 26, 27, and 29
- G. JBuilder Tutorial
- H. NetBeans Tutorial
- I. Eclipse Tutorial
- J. Tutorial for MySQL
- K. Tutorial for Oracle
- L. Tutorial for Microsoft Access
- M. Tutorial for Tomcat
- N. Creating Shortcut for Java Applications on Windows
- O. Supplemental Case Study for Chapter 10: Design a GenericMatrix Class
- P. Creating Generic Types (JDK 1.5)
- Q. Enumerated Types (JDK 1.5)
- R. Semaphores (JDK 1.5)

## Acknowledgments

I would like to thank Ray Greenlaw, Chuck Shipley, and my colleagues at 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. Their enthusiastic support has contributed to the success of my Java series.

This book was greatly improved thanks to outstanding reviews by James Chegwidden of Tarrant County College, Dan Lipsa of Armstrong Atlantic State University, Vladan Jovanovic of Georgia Southern University, Kenrick Mock of the University of Alaska—Anchorage, Ronald F. Taylor of Wright State University, and Lixin Tao of Pace University.

It is a great pleasure and privilege to work with the legendary computer science team at Prentice Hall. I would like to thank Alan Apt, Toni Holm, Patrick Lindner, Sarah Parker, Camille Trentacoste, John Keegan, Xiaohong Zhu, Pamela Shaffer, Barrie Reinhold, Barbara Taylor-Laino, Meredith Maresca, and their colleagues for organizing, managing, and promoting this project, and to Robert Milch for copy editing.

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

Y. DANIEL LIANG  
 liang@armstrong.edu  
 www.cs.armstrong.edu/liang