JAVA™ SOFTWARE SOLUTIONS

Foundations of Program Design

JOHN LEWIS
WILLIAM LOFTUS

Java™ Software Solutions

Foundations of Program Design

John Lewis
Villanova University

William Loftus
WPL Laboratories, Inc.

ADDISON-WESLEY

An imprint of Addison Wesley Longman, Inc.

Reading, Massachusetts • Harlow, England • Menlo Park, California
Berkeley, California • Don Mills, Ontario • Sydney • Bonn • Amsterdam
Tokyo • Mexico City

Editor-in-Chief	Lynne Doran Cote
Associate Editor	Deborah Lafferty
Production Manager	Karen Wernholm
Production Editor	Amy Willcutt
Marketing Manager	Tom Ziolkowski
Compositor	Michael and Sigrid Wile
Technical Artist	George Nichols
Copyeditor	Roberta Lewis
Text Design	Ron Kosciak
Indexer	Nancy Fulton
Proofreading	Phyllis Coyne et al.
Cover Designer	Diana Coe

Library of Congress Cataloging-in-Publication Data

Lewis, John, Ph.D.
Java software solutions : foundations of program design / John Lewis, William Loftus.
p. cm.
Includes index.
ISBN 0-201-57164-1
1. Java (Computer program language) 2. Object-oriented programming (Computer science) I. Loftus, William. II. Title.
QA76.73.J38L49 1998
005.13'3--dc21 97-19400
CIP

Access the latest information about Addison-Wesley titles from our World Wide Web site: http://www.awl.com/cseng

Reprinted with corrections, January 1998.

6 7 8 9 10-MA-01009998

Preface

We have designed this text for use in a first course in programming using the Java language. It serves as an introduction to computer science and forms a foundation for pursuing advanced computing topics. Our goal is to make students comfortable with object-oriented concepts so that they will be well-prepared to design and implement high-quality object-oriented software.

This text was formed out of our combined experiences with real-world programming and classroom teaching of Java. We have written this text from the ground up, with an object-oriented Java approach always in mind. When Java first emerged in mid-1995, most of the attention was focused on its applets and glitzy Web effects. Over time, people have come to realize its larger benefits as a powerful object-oriented language that is well-designed and pedagogically sound. We have discovered that students respond better, faster, and more enthusiastically to computing concepts when they are explained through Java.

In response to a strong initial interest, this text was first published in a preliminary version. Since then we have made several improvements to the text, rearranging topics to provide maximum versatility and adding more examples to help students better grasp important concepts. This edition has also been updated to fully embrace Java 1.1. This new version of Java provides many improvements over the earlier version, including a significant improvement to the GUI event model.

Object-Oriented Coverage

We introduce objects early in the text and consistently reinforce their use throughout. We have found that students find object-oriented concepts highly intuitive if they are presented in a clear, careful way. Introductory programmers can success-

fully master concepts like inheritance and polymorphism when these ideas are discussed in a straightforward and thorough manner.

The term object-oriented software development implies that the approach is oriented around objects, yet some people advocate postponing the introduction of objects until after many traditional procedural techniques are covered. Our view is that as soon as a design gets sophisticated enough to deserve multiple methods, it should use objects with methods in them. Methods should never be taught independent of their role in an object. We believe that educating students in object-oriented design will prepare them to be better programmers regardless of the language used.

GUI Coverage

We have experimented with a variety of approaches and have concluded that our students should not be asked to develop graphical user interfaces in Java too early. Introducing GUIs prior to a thorough coverage of classes, interfaces, and inheritance requires too many vague and misleading side discussions. We still cover applets, graphics, and animation early, but defer event-based interaction until suitable foundation material has been established.

The Four Cornerstones of the Text

This text is based upon four basic ideas that we believe make for a sound introductory text.

- *True object-orientation.* A true object-oriented text must do more than mention objects early. In this text, every situation and example reinforces the design principles of object-oriented programming. We establish as a fundamental guideline that the class that contains the `main` method should contain no additional methods; if other functionality is needed, it is provided through other classes and objects. This guideline is applied in all programs as soon as objects are introduced. (See the `CD_Collection` example in Chapter 4.)
- *Sound software engineering.* Students should be exposed to software engineering principles early in order to be prepared to develop high-quality software in the future. Software engineering concepts are integrated throughout the text and repeatedly reinforced so that students learn their importance from the start. For example, design and process issues are introduced in Chapter 3 and revisited in examples throughout the text. Furthermore, Chapters 11 and 15 are devoted to software engineering issues.
- *Integrated graphics.* Modern software systems are graphical. Introductory programming courses should cover graphics and graphical user interfaces. Various examples in this text, as early as the `No_Parking` applet in Chapter 2, use graphics to motivate and engage students. Furthermore, we

devote Chapter 7 to a complete investigation of basic Java graphics and Chapter 10 to GUIs and related topics. We introduce GUIs carefully, after students can appreciate the concepts of event-driven programming.

- *Balanced examples.* A text must contain a strong balance of smaller and larger examples. Smaller examples establish a foundation for students, whereas larger examples provide them with a more realistic context. We have intertwined small, readily understandable examples with larger, practical ones to give students and faculty a variety of examples to explore. We also balance the use of applications and applets throughout the text in order to give students a strong foundation in both approaches.

Paths Through the Text

This book is designed to be flexible, so that professors can tailor its presentation to the needs of their students. Professors can take a variety of different paths through the text; these paths include four major topics—object-oriented development, graphics and GUIs, software engineering, and Java language features. The initial chapters should be covered in the designated order, as they form the foundation on which to explore these topics.

Chapter 1 (Computer Systems) presents a broad overview of computing topics. It establishes some terminology concerning hardware, networks, and the World-Wide Web. Depending on the background of the student, this chapter can be covered quickly or left for outside reading. Chapter 2 (Software Concepts) begins the exploration of software development and introduces the concepts underlying the object-oriented approach. Students with previous software development exposure may only need to focus on portions of Chapter 2 as needed. Chapter 3 (Program Elements) provides just enough low-level detail, including basic control flow, in order to make the exploration of objects concrete.

Chapter 4 (Objects and Classes) is the springboard for the rest of the book. It describes how to define objects using classes and the methods and data that they contain. At this point the instructor has wide latitude in choosing the topics that will follow. Chapter 5 (More Programming Constructs) can be covered immediately to fill in additional low-level details, or it can be deferred to a later point. A more traditional course flow might also include Chapter 6 (Objects for Organizing Data) with its emphasis on arrays.

The remaining chapters can be organized in a variety of different ways based upon the needs of the instructor. Those instructors who want to emphasize object-oriented development can follow Chapter 4 with Chapter 8 (Inheritance) and Chapter 9 (Enhanced Class Design). The object-oriented issues should be covered prior to introducing the graphical user interface material in Chapter 10, although the basic graphics content of Chapter 7 can be covered any time after Chapter 4. A software engineering track can be followed by covering Chapters 11 and 15 (Software Development Process I and II) after the object-oriented material. To

emphasize the Java language features, instructors can follow Chapters 4, 5, and 6 with Chapters 8, 9, and 14 (Advanced Flow of Control). We invite instructors to experiment with the ordering of chapters to best meet their own course needs.

Pedagogical features

This text contains numerous pedagogical features that help make the material more accessible to students. Some of the features we use are listed below:

- *Key Concepts.* The Key Concept designation is used throughout the book to draw special attention to fundamental ideas and important design guidelines.
- *In-Depth Focus boxes.* These boxes appear in several places throughout the text and provide a tiered coverage of material. They allow more advanced students to challenge their knowledge of the subject without overwhelming others. Instructors may choose to cover or skip this feature without any loss of continuity.
- *Code Callouts.* Blue type is used to call out and annotate important parts of the code. The second color allows students to better understand the code as they read through it.
- *Problem sets.* Each chapter of the book concludes with a set of problems, separated into three categories:
  - Self-Review Questions and Answers. These short-answer questions review the fundamental ideas and terms established in the chapter. 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 problem sets.
  - Exercises. These intermediate problems probe the underlying issues discussed in the chapter and integrate them with concepts covered in previous chapters. While they may deal with code, they do not involve any online activity.
  - Programming Projects. These consist of more involved problems that require design and implementation of Java programs. The projects vary widely in level of difficulty.
- *Java reference material.* The appendices contain a significant amount of language reference material. We have placed this material in appendices so that more of the text can focus on the important software concepts. Students can reference these appendices as needed throughout the course to learn more details of the Java language.
- *Java style guidelines.* Appendix G contains a proposed set of programming style guidelines. These guidelines are followed in the examples throughout the text.

- *Graphical design notation.* The object-oriented designs in the text are presented with a simple graphical notation. This allows students to read and use a design notation similar to professional development models.

Conventions

We use various conventions for indicating different types of material in the text. Important words and phrases are emphasized in *italics* on their first use. Code is presented in a monospaced font:

```
void cube (int num) {
   System.out.println ("The cube is " + (num*num*num));
} // method cube
```

and code elements, such as `cube`, maintain the code font in the text. Output is presented in a monospaced font surrounded by a colored box:

```
The cube is 9
```

Specific syntax of individual programming statements are shown in shaded boxes:

```
return-type method-name ( parameter-list ) {
   statement-list
}
```

In the sample run of a program, user input is shown in color:

```
> java Average
Enter a number (-1 to quit): 90
Enter a number (-1 to quit): 80
Enter a number (-1 to quit): 70
Enter a number (-1 to quit): -1
The average is 80
```

Pseudocode is presented in a script font:

```
prompt for and read the grade
  while (grade does not equal -1) {
  increment count
  sum = sum + grade;
  prompt for another grade
  read next grade
}
average = sum / count;
print average
```

Supplements

This book comes with a large variety of supplemental materials to assist in course preparation and execution. Links to all of the supplements can be found on the book's official Web site at `http://www.awl.com/cseng/author/lewis/java`. In addition to the supplements listed below, this site contains all examples from the book and additional Java examples not found in the book.

- *Instructor's Manual.* A manual has been created to assist professors in course preparation. It contains strategy suggestions for presenting material, answers to text exercises, solutions to selected programming projects, and a collection of potential test questions and answers. To obtain a copy of the Instructor's Manual, please contact your local Addison-Wesley sales representative.
- *Laboratory Manual.* A series of independent exercises support curricula that use a closed lab approach. Instructors can choose from a variety of labs, covering material found in each chapter of the text. The labs overlap to reflect the various ways that an instructor can approach the book. In addition to use in the laboratory environment, the lab exercises may also be assigned as outside work.
- *Integrated Web Presentation.* These Web pages allow an instructor to interactively present course notes, examples, and executable code entirely through a Web browser. At the instructor's discretion, the material can then be made available to students for further review at their own pace.
- *Transparency Masters.* Overhead slides are available for those who choose not to use the Integrated Web Presentation. Slides may be obtained in either Microsoft PowerPoint format or PostScript.

Acknowledgments

The creation of this text was an effort that extends well beyond the authors. If we have succeeded in our goals, it is largely due to the support we received from many sources.

First of all, we greatly appreciate the students who have participated in the courses in which preliminary versions of this text were used. Their feedback and suggestions have been quite helpful in the process of refining the book's content and presentation.

Lynne Doran Cote and Debbie Lafferty at Addison-Wesley have been outstanding in their editorial support and encouragement. Amy Willcutt was amazingly helpful and accommodating during the final production of the text, with the support of Karen Wernholm. Tom Ziokowski, Michael Hirsch, and Stacy Treco provided important insight and direction. Roberta (Bobbie) Lewis, of Lewis Editorial Services, was a pleasant and meticulous copyeditor. We appreciate their support of our vision for this book and their desire for quality above all else.

Many thanks go to our reviewers, listed below, who provided important, constructive comments and suggestions. They found numerous ways to improve the quality of the text and were never shy about expressing their opinion. Any errors that still exist in the book are solely the responsibility of the authors, as we can never seem to stop making changes.

Christopher Haynes	Indiana University
Lawrence Osborne	Lamar University
B. Ravikumar	University of Rhode Island
David Riley	University of Wisconsin, LaCrosse
Vijay Srinivasan	JavaSoft, Sun Microsystems Inc.
Shengru Tu	University of New Orleans
John J. Wegis	JavaSoft, Sun Microsystems Inc.
David Wittenberg	Brandeis University

Thanks also go to the many informal reviewers who have provided valuable feedback. Chief among them is Dan Joyce of Villanova University, who was instrumental in helping us revise our initial approach and who provided guidance through multiple revisions. Paul Gormley also provided significant and helpful comments on the content of the text.

Special thanks go to Pete DePasquale at Villanova University. He has been a tremendous help in many areas, including the development of Appendix O, the creation of exercises, and overall review. His assistance has been invaluable.

Many other people have helped in various ways. They include Ken Arnold, Bob Beck, Alan Dellinger, Tom DiSessa, Dan Hardt, John Loftus, Bob Pollack, Tim Ryan, Brent Schwartz, Ken Slonneger, Joe Tursi, and Mahesh Vanavada. Our apologies to anyone we may have forgotten.

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.

The faculty in the Department of Computing Sciences at Villanova University and the staff at WPL Laboratories, Inc. have supported us both throughout this process. Their support is greatly appreciated.

Thanks also go to the following: Sun Microsystems (the network *is* the computer), FedEx (it often had to be there overnight), WaWa (open 24 hours, including holidays), Dominos (they deliver), Diet Coke (just for the taste of it), New Orleans (especially the House of Blues), sleep (we've read about this), coffee (the elixir of life), Altoids (curiously strong), a helpful student (for the goat), and the couch of science (the seat of inspiration).

Most importantly, thanks go to our wives. John thanks his wife Sharon for her love and understanding throughout this project, and for distracting him when he needed it. Bill thanks his wife Veena, for her undying love and support, his son Isaac, for his inspirational story "The Golden Mask," and his daughter Devi, for teaching him how to dress.

John Lewis
William Loftus

Contents