



75+ HOURS TO COMPLETE

JAVA™

Complete Course

in Programming & Problem Solving



LAMBERT & OSBORNE



986116

JAVATM

Complete Course

in Programming & Problem Solving

Dr. Kenneth A. Lambert

Washington & Lee University

Dr. Martin Osborne

Western Washington University

江苏工业学院图书馆
藏书章

VISIT US ON THE INTERNET

www.swep.com



South-Western Educational Publishing

an International Thomson Publishing company ITP®

www.thomson.com

Cincinnati • Albany, NY • Belmont, CA • Bonn • Boston • Detroit • Johannesburg • London • Madrid
Melbourne • Mexico City • New York • Paris • Singapore • Tokyo • Toronto • Washington

To Nathaniel—Ken
To Tess—Martin

Library of Congress Cataloging-in-Publication Data

Lambert, Kenneth (Kenneth A.)

Java: complete course in programming and problem solving /
Kenneth A. Lambert, Martin Osborne.

p. cm.

Includes index.

ISBN 0-538-68707-X (hardbound). — ISBN 0-538-68711-8 (softcover)

1. Java (Computer program language) I. Osborne, Martin

QA76.J38L355 2000

005.13'3—dc21

98-35332
CIP

Managing Editor:	Carol Volz
Project Manager:	Dave Lafferty
Consulting Editor::	Custom Editorial Productions, Inc.
Marketing Manager:	Steve Wright & Larry Qualls
Design Coordinator:	Mike Broussard
Production:	Custom Editorial Productions, Inc.

Copyright © 2000

by SOUTH-WESTERN EDUCATIONAL PUBLISHING

Cincinnati, Ohio

ALL RIGHTS RESERVED

The text of this publication, or any part thereof, may not be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, storage in an information retrieval system, or otherwise, without the prior written permission of the publisher.

The names of commercially available software mentioned herein are used for identification purposes only and may be trademarks or registered trademarks of their respective owners. South-Western Educational Publishing disclaims any affiliation, association, connection with, sponsorship, or endorsement by such owners.

ISBN: 0-538-68707-X (hard cover)
0-538-68771-8 (soft cover)

1 2 3 4 5 D 02 01 00 99 98

Printed in the United States of America

I(T)P®

International Thomson Publishing

South-Western Educational Publishing is a division of International Thomson Publishing, Inc. The ITP registered trademark is used under license.

BreezyGUI® is a registered trademark of Brooks/Cole Publishing, a division of International Thomson Publishing, Inc.

PREFACE

This text is intended for a first course in programming and problem solving. We want students to focus on traditional topics in computer science, while writing object-oriented programs with graphical user interfaces in Java. Thus, the text covers seven major aspects of computing:

1. **Programming Basics.** This deals with the basic ideas of problem solving with computers, primitive data types, control structures, and methods.
2. **Data and Information Processing.** Fundamental data structures are discussed. These include strings, arrays, and files.
3. **Object-Oriented Programming.** OOP is today's dominant programming paradigm. All the essentials of this subject are covered.
4. **Graphical User Interfaces and Event-Driven Programming.** Many texts at this level cling to what has now become an antiquated mode of programming—character-based terminal I/O. The reason is simple. GUIs and event-driven programming are too complex for beginning students. In this text, we overcome the barrier of complexity in the manner explained below.
5. **Software Development Life Cycle.** Rather than isolate software development techniques in separate lessons, our text deals with them throughout in the context of numerous case studies.
6. **Graphics.** Our text explores problem solving with simple graphics. This includes drawing simple shapes, representing data graphically, and implementing a rudimentary sketching program.
7. **Networking.** We introduce the programming of Web pages and applets.

Early, Easy GUIs with BreezyGUI®

Every CS1 Java text faces a dilemma: either do terminal I/O and look like a C++ text, or do graphical user interfaces and overwhelm the reader with the details of Java's Abstract Windowing Toolkit. To overcome this dilemma, our text comes with a software package, **BreezyGUI**, which simplifies the programming of graphical user interfaces. **BreezyGUI** insulates students from the complex details of setting up window objects and managing interface events. Thus, students can use GUIs without being overwhelmed and distracted from the basic business of software development—algorithm design and factoring code into classes. Every example program in the first 12 lessons is GUI-based and uses **BreezyGUI**. The mystery behind the **BreezyGUI** package is removed in the final lesson of the text, where we introduce the details of Java's Abstract Windowing Toolkit and delegation event model.

Focus on Traditional Computer Science Topics

Many introductory Java books succumb to the temptation to focus on the popular features of Java for Web-based programs, such as applets, threads, client/server network applications, and multimedia. We believe that these are actually advanced topics, which presuppose a principled introduction to the field. The example programs in the first 11 lessons of our book are stand-alone Java applications. Lesson 12 introduces HTML programming and applets, which allow Java programs to run in Web browsers. Because all of our applications are GUI-based, the transition from applications to applets is straightforward.

Just-in-Time, Multiparadigm Approach to Problem Solving

At one time there was a movement in computer science texts to introduce user-defined procedures as early as possible. Many texts are now supplanting this approach with another one: introduce user-defined classes as early as possible. Both approaches overlook the fact that procedures and classes are mechanisms for structuring code, and as such they are best introduced when students start working with problems that call for these organizational tools. Thus, the early lessons of our book (2 through 4) emphasize calculations, control constructs, and algorithms. User-defined methods arrive in Lesson 5 and user-defined classes in Lesson 7. There they are needed, and their benefits are appreciated.

Case Studies and the Software Life Cycle

This text contains numerous case studies. These are complete Java programs, ranging from the simple to the substantial. To emphasize the importance and reality of the software development life cycle, case studies are always presented in the framework of a user request, analysis, design, and implementation, with well-defined tasks performed at each stage. Some case studies are carried through several lessons or extended in end-of-lesson programming projects.

Alternative Paths Through the Text

We have organized the text to satisfy different time frames and topic preferences. We recommend starting with the first three lessons, which provide CS background; explain how to run a first Java program; and explore the basics of syntax, semantics, and debugging. After the first three lessons have been covered, the following alternatives suggest themselves:

1. Those who cannot resist doing applets should skip to Lesson 12 and then return to Lesson 4.
2. Those who cannot resist doing graphics should do Lessons 4 and 5, then skip to Lesson 10, and return to Lesson 6.
3. Those who cannot resist doing files should do Lessons 4–6 then skip to Lesson 11, and return to Lesson 7.
4. Those who want to cover user-defined classes but omit inheritance should skip Lesson 9.

Of course, the best way to use this text is simply to go through it.

Reports of Errors and Updates

We have made every effort to produce an error-free text, although this cannot be guaranteed with certainty. We assume full responsibility for all errors or omissions. Readers are encouraged to report errors to klambert@wlu.edu. A listing of errata, should they be discovered, and other information about the book will be posted on the author's Web site, <http://www.wlu.edu/~lambertk/hsjava/index.html>.

Acknowledgments

We would like to take this opportunity to thank those who in some way contributed to the completion of this text. Several reviewers offered constructive comments during various phases of this project. They include:

Mark Ciampa, Volunteer State Community College, Gallatin, TN

Fred Bartels, Rye Country Day School, Rye, NY

Derek Hodgkins, New Hampshire Community Technical College, Manchester, NH

Lily Hou, Carnegie-Mellon University, Pittsburgh, PA

Tsang-Ming Jiang, University of Alaska, Fairbanks, AK

Gail Miles, Lenoir-Rhyne College, Hickory, NC

Arland J. Richmond, Computer Learning Center, Somerville, MA

Christopher Starr, College of Charleston, Charleston, SC

Hoyt D. Warner, Western New England College, Springfield, MA

Lee Wittenberg, Kean University, Union, NJ

Winnie Yu, Southern Connecticut State University, New Haven, CT

John Zelle, Drake University, Des Moines, IA

Several other people deserve mention because, without their expertise, this text would not exist:

Cat Skintik of Custom Editorial Productions, Inc., Developmental Editor. Cat worked through not only the text but also all of the ancillaries and program code, testing all of the programs, taking the quizzes, and offering helpful commentary.

Jim Reidel, Valerie Brandenburg, and Cindy Lanning of Custom Editorial Productions, Inc., for the production and layout of this book.

John Wills, Partnerships Manager. Among other things, John negotiated the agreement that resulted in the compiler that is bundled with the text.

Dave Lafferty, Project Manager. Dave drew up the blueprints for a complete teaching package, from text to workbook to instructor's CD, and kept the project on target.

Carol Volz, Managing Editor. Carol focused our eyes on the need for this text and worked out the arrangements for placing it in South-Western's computer education listing.

Finally, we are grateful to our wives and children for giving us the time and support to develop this text.

**Kenneth A. Lambert
Martin Osborne**

How to Use this Text

What makes a good computer programming text? Sound pedagogy and the most current, complete materials. That is what you will find in the new *Java: Complete Course in Programming and Problem Solving*. Not only will you find an inviting layout, but also many features to enhance learning.

Objectives—

Objectives are listed at the beginning of each lesson, along with a suggested time for completion of the lesson. This allows you to look ahead to what you will be learning and to pace your work.

Enhanced Screen

Shots—Screen shots now come to life on each page.

Program Code

Examples—Many examples of program code are included in the text to illustrate concepts under discussion.

LESSON

X

USER-DEFINED METHODS

OBJECTIVES

Upon completion of this lesson, you should be able to:

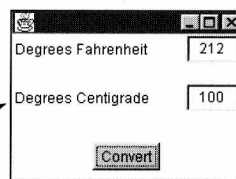
- Design and implement methods.
- Discuss how methods use parameters and return values.

Estimated Time: 3 hours

User-Defined Methods

The programs we have written so far have been short and simple, but soon we will tackle problems of greater complexity. To manage the complexity, we can apply a strategy that has proven successful in many areas of human endeavor—divide and conquer.

FIGURE X-X
The proposed interface



The method `Math.pow (x, y)` raises `x` to the power of `y`. The following code segment displays the first ten powers of 2 on separate lines in a text area named `output`:

```
int expo, data;
expo = 1;
while (expo <= 10){
    data = Math.pow (2, expo);
    output.append (data + "\n");
}
```

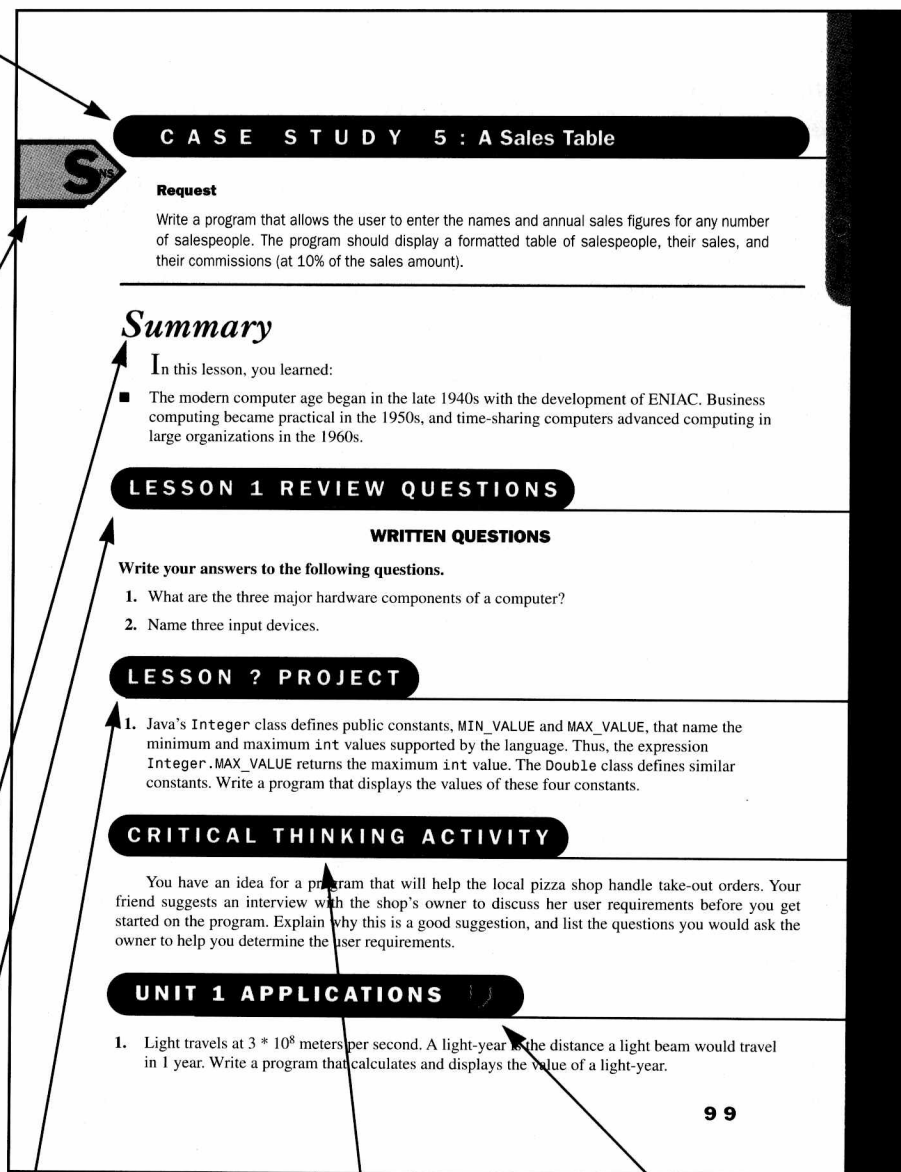
How to Use this Text

Case Studies—Case studies present Java program solutions to specific user requests and show the analysis, design, and implementation stages of the software development life cycle.

SCANS (Secretary's Commission on Achieving Necessary Skills)—The U.S. Department of Labor has identified the school-to-careers competencies (resources, interpersonal skills, information, systems, and technology) and foundation skills (basic skills, thinking skills, and personal qualities) are identified in Case Studies and Projects throughout the text. More information on SCANS can be found on the *Electronic Instructor*.

Summary—At the end of each lesson, you will find a summary to help you complete the end-of-lesson activities.

Review Questions—Review material at the end of each lesson and each unit enables you to prepare for assessment of the content presented.



CASE STUDY 5 : A Sales Table

Request

Write a program that allows the user to enter the names and annual sales figures for any number of salespeople. The program should display a formatted table of salespeople, their sales, and their commissions (at 10% of the sales amount).

Summary

In this lesson, you learned:

- The modern computer age began in the late 1940s with the development of ENIAC. Business computing became practical in the 1950s, and time-sharing computers advanced computing in large organizations in the 1960s.

LESSON 1 REVIEW QUESTIONS

WRITTEN QUESTIONS

Write your answers to the following questions.

- What are the three major hardware components of a computer?
- Name three input devices.

LESSON 2 PROJECT

- Java's Integer class defines public constants, MIN_VALUE and MAX_VALUE, that name the minimum and maximum int values supported by the language. Thus, the expression Integer.MAX_VALUE returns the maximum int value. The Double class defines similar constants. Write a program that displays the values of these four constants.

CRITICAL THINKING ACTIVITY

You have an idea for a program that will help the local pizza shop handle take-out orders. Your friend suggests an interview with the shop's owner to discuss her user requirements before you get started on the program. Explain why this is a good suggestion, and list the questions you would ask the owner to help you determine the user requirements.

UNIT 1 APPLICATIONS

- Light travels at 3×10^8 meters per second. A light-year is the distance a light beam would travel in 1 year. Write a program that calculates and displays the value of a light-year.

99

Lesson Projects—End-of-lesson hands-on application of what has been learned in the lesson allows you to actually apply the techniques covered.

Critical Thinking Activity—Each lesson and each unit review gives you an opportunity to apply creative analysis to situations presented.

End-of-Unit Applications—End-of-unit hands-on applications of concepts learned in the unit provides opportunity for a comprehensive review.

Explore the Flavor of Java!

With these exciting new products from South-Western!

Our new Java programming books offer everything from beginning, to intermediate, to advanced courses to meet your programming needs.

- **NEW! *Java Complete Course in Programming and Problem Solving*** by Lambert and Osborne is the most comprehensive instructional text available for learning Java. It contains 75+ hours of instruction on the most widely used beginning-through-advanced features of Java. Covers Java for both Windows and Macintosh.

Student book, hard cover	0-538-68707-X
Student text-workbook/data CD-ROM package, soft cover	0-538-68708-8
Activities Workbook	0-538-68710-X
Electronic Instructor CD-ROM package	0-538-68709-6

- **NEW! *Java: Introduction to Programming*** by Knowlton covers the beginning-through-intermediate features of Java in 35+ hours of instruction. The text is available in hard or soft cover and is for the Windows version of Java only.

Student book, hard cover	0-538-68565-4
Student book/3.5" template disk package, soft cover	0-538-68772-X
Activities Workbook	0-538-68571-9
Electronic Instructor CD-ROM package	0-538-68557-3
Student book, hard cover/Microsoft Visual J++ package	0-538-68774-6
Student book/3.5" template disk package, soft cover with Microsoft Visual J++ package	0-538-68773-8

- **NEW! *Java: Programming Basics for the Internet*** by Barksdale and Knowlton, et al., gives the user a quick introduction to the beginning-through-advanced features of Java. It contains 15+ hours of instruction; the emphasis is on applets and activities.

Student Text-Workbook	0-538-68012-1
Student Text-Workbook/Microsoft Visual J++ package	0-538-68564-6
Instructor's Manual (online)	0-538-68013-X

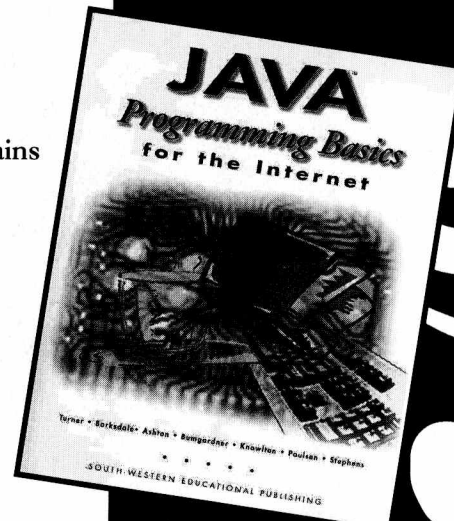
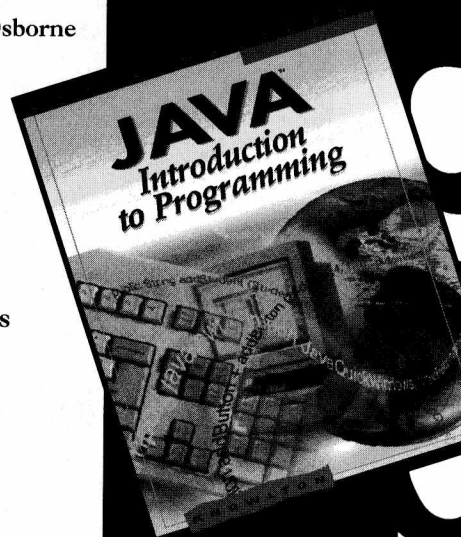
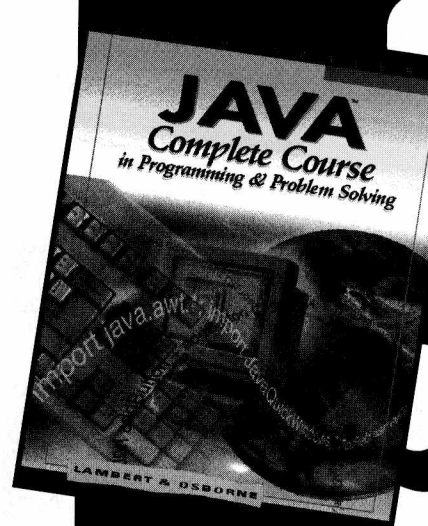
A new feature available for these products is the *Electronic Instructor*, which includes a printed Instructor's manual and a CD-ROM. The CD-ROM contains tests, lesson plans, all data solutions files, and more! Also, ask about our ProgramPaks for compiler software bundles!

For information call 1-800-354-9706.



South-Western
Educational Publishing

Join Us On the Internet
www.swep.com





CONTENTS

UNIT 1

GETTING STARTED WITH JAVA

1

2 Lesson 1: A Brief History of Computer Programming

History of Computers	2
Computer Architecture	3
Programming Languages	5
Software Development Process	6
Object-Oriented Programming	9
CS Capsule: The ACM Code of Ethics and Intellectual Property	11
Summary	13

15 Lesson 2: A First Java Program

Why Java?	15
The Java Virtual Machine and Byte Code	16
Case Study: Request, Analysis, and Design for the First Program	16
Writing the Program	18
Language Elements	19
Interpreting the Program	20
Overview of Editing, Compiling, and Running a Program	24
Creating and Running the First Program	25
Formatting a Program and Comments	28
Programming Errors	29
Illustration of Syntax Errors	30
Illustration of Run-Time Errors	31
Illustration of Logic Errors	33
Debugging	34
Applets and Stand-alone Programs	35
CS Capsule: Intrusive Hacking and Viruses	35
Summary	36

40 Lesson 3: Java Basics

Case Study 1: Area of a Circle	40
Case Study 2: Income Tax Calculator	42

CS Capsule: Binary Representation of Information and Computer Memory	43
Numeric Data Types and Numeric Literals	47
Variables	48
Naming Other User-Defined Symbols	49
Expressions	49
Mixed-Mode Arithmetic	51
Tester Programs	52
BreezyGUI: Layout, Objects, and Methods	53
Strings	57
Case Study 3: Vital Statistics	61
Design, Testing, and Debugging Hints	62
Summary	63

67 Lesson 4: Control Statements

A Visit to the Farm	67
The if and if-else Statements	68
Relational Operators and Their Precedence	70
Case Study 1: Circle Area and Radius	71
The while Statement	73
Case Study 2: Count the Divisors	77
Case Study 3: Fibonacci Numbers	78
Nested if Statements	80
Data Validation and Robust Programs	83
Case Study 4: Making CircleAreaAndRadius Robust	83
BreezyGUI: Text Areas and Formatted Output	84
Case Study 5: A Sales Table	87
CS Capsule: Artificial Intelligence, Robots, and Softbots	89
Design, Testing, and Debugging Hints	89
Summary	90

94 Unit 1 Review

UNIT 2

METHODS, DATA TYPES, AND CLASSES

97

98 Lesson 5: User-Defined Methods

User-Defined Methods	98
----------------------	----

Parameters and Return Values	101
Scope and Lifetime of Variables	104

Preconditions and Postconditions	106
CS Capsule: Function-Oriented Programming	107
Case Study: Tally Grades	108
Finding the Location of Run-Time Errors	116
Other Implementation Strategies	117
Recursion	117
Design, Testing, and Debugging Hints	120
Summary	121

126 Lesson 6: More Operators, Control Statements, and Data Types

Operators	126
Control Statements	132
The Math Class	136
Data Types	137
Constants	142
Case Study 1: Metric Conversion	143
Case Study 2: Dice Rolling	144
Strings Revisited	145
Case Study 3: Palindromes	149
CS Capsule: Data Encryption	151
Design, Testing, and Debugging Hints	151
Summary	152

156 Lesson 7: User-Defined Classes

Overview of Classes and Objects	156
A Student Class	157
Editing, Compiling, and Testing the Student Class	164
BreezyGUI: Menus and the Title	165
Case Study: Student Test Scores	166
The Static Modifier	172
Restriction on the Use of the messageBox Method	175
Constructors	175
Primitive Types, Reference Types, and the null Value	177
Copying Objects	178
Comparing Objects for Equality	179
CS Capsule: Reliability of Software Systems	180
Design, Testing, and Debugging Hints	181
Summary	181

185 Unit 2 Review

UNIT 3

ARRAYS, INHERITANCE, AND GRAPHICS

189

190 Lesson 8: Arrays, Searching, and Sorting

Arrays	190
Parallel Arrays	200
Two-Dimensional Arrays	201
Arrays and Methods	203
BreezyGUI: Checkboxes, Radio Buttons, Scrolling Lists, and Choice Lists	206
Case Study 1: Polynomial Evaluator	214
Case Study 2: Student Test Scores Again	221
The Model/View Pattern	228
Design, Testing, and Debugging Hints	235
Summary	236

242 Lesson 9: Inheritance, Abstract Classes, and Polymorphism

Introduction	242
Implementing a Simple Shape Hierarchy	243
Using the Shape Classes	249
Extending the Shape Hierarchy	250
Arrays of Shapes	253
Shapes as Parameters and Return Values	255
An Employee Hierarchy	258

Case Study: The Painter's Friend	260
Object-Oriented Analysis and Design Guidelines	270
Summary	271

273 Lesson 10: Simple Graphics

The Conceptual Framework for Computer Graphics	273
Case Study 1: Drawing Text at Different Positions	278
Color	280
Tracking the Mouse	281
Case Study 2: A Simple Sketching Program	282
Transient and Refreshable Images	283
Defining and Using a Geometric Class	284
Case Study 3: Dragging Circles	286
Text Properties	291
Design, Testing, and Debugging Hints	292
Summary	292

296 Unit 3 Review

300 Lesson 11: Files

Secondary Storage	300
File Classes	301
File Input	302
Case Study 1: A Text Analyzer	308
File Output	311
Case Study 2: Employees and Payroll	312
Data Input and Output Streams	319
Serialization and Object Streams	321
Terminal Input and Output	323
File Dialogs	324
CS Capsule: Programming Language Translation	326
Design, Testing, and Debugging Hints	327
Summary	327

330 Lesson 12: Introduction to HTML and Applets

Hypertext, Hypermedia, and the World Wide Web	330
Hypertext Markup Language	331
Simple Text Elements	335
Character-Level Formatting	336
Lists	337
Linking to Other Documents	340
Multimedia	342
Tables	344
Applets	346
Case Study 1: Fahrenheit to Centigrade as an Applet	347
Compiling and Running an Applet	349
Differences Between Applets and Applications	350
Case Study 2: A Game of Tic-Tac-Toe	350
Design, Testing, and Debugging Hints	355
Summary	355

357 Lesson 13: The Abstract**Windowing Toolkit**

The AWT Philosophy	357
Conversion Program Implemented with GBFrame	358

Conversion Program Implemented with AWT	360
GUI Components	368
Layouts	372
All About Events	382
Dialogs	388
The Model/View/Controller Pattern	391
Case Study: A Color Meter Application	392
Applets and the AWT	397
Summary	398

401 Unit 4 Review**404 Appendix A: Java Resources and Environments****428 Appendix B: Reserved Words****429 Appendix C: Operator Precedence****431 Appendix D: ASCII Character Set****432 Appendix E: Number Systems****435 Appendix F: Java Exception Handling****437 Appendix G: Java Packages****438 Appendix H: BreezyGUI****443 Glossary****456 Index**

GETTING STARTED WITH JAVA

UNIT 1

lesson 1

2 hr.

A Brief History of
Computer Programming

lesson 2

3.5 hr.

A First Look at Java

lesson 3

4 hr.

Java Basics

lesson 4

4 hr.

Control Structures



Unit 1 Estimated Time: 13.5 hours



A BRIEF HISTORY OF COMPUTER PROGRAMMING

OBJECTIVES

Upon completion of this lesson, you will be able to:

- Give a brief history of computers.
- Describe how hardware and software make up computer architecture.
- Discuss the evolution of programming languages.
- Describe the software development process.
- Discuss the fundamental concepts of object-oriented programming.

 Estimated Time: 2 hours

History of Computers

ENIAC, built in the late 1940s, was one of the world's first computers. It was a large, stand-alone machine that filled a room and used more electricity than all the houses on an average city block. ENIAC contained hundreds of miles of wire and thousands of heat-producing vacuum tubes. The mean time between failures was less than an hour, yet because of its fantastic speed, when compared to hand-operated electro-mechanical calculators, it was immensely useful. To read more about the ENIAC and see photos of early computers, contact the following site by using your Web browser: <http://ftp.arl.mil/ftp/historic-computers>.

In the early 1950s, IBM sold its first business computer. At the time, analysts estimated that the world would never need more than ten such machines, yet its awesome computational power was a mere 1/200 of the typical 200-megahertz Pentium personal computer purchased for about \$1000 in 1998.

The first computers could perform only a single task at a time. Input and output were handled by such primitive means as switches, punch cards, and paper tape.

In the 1960s, time-sharing computers, costing hundreds of thousands and even millions of dollars, became popular at organizations large enough to afford them. Even back then, computers were so much faster that 30 people could work on such a computer simultaneously without loss of computing power. Each person sat at a Teletype console connected by wire to the computer. By making a connection through the telephone system, Teletype consoles could be placed at a great distance from the computer. The Teletype was a primitive device by today's standards. It looked like an electric typewriter with a large roll of paper attached. Keystrokes entered at the keyboard were transmitted to the computer, which then echoed them back on the roll of paper. Output from the computer's programs was also printed on this roll.

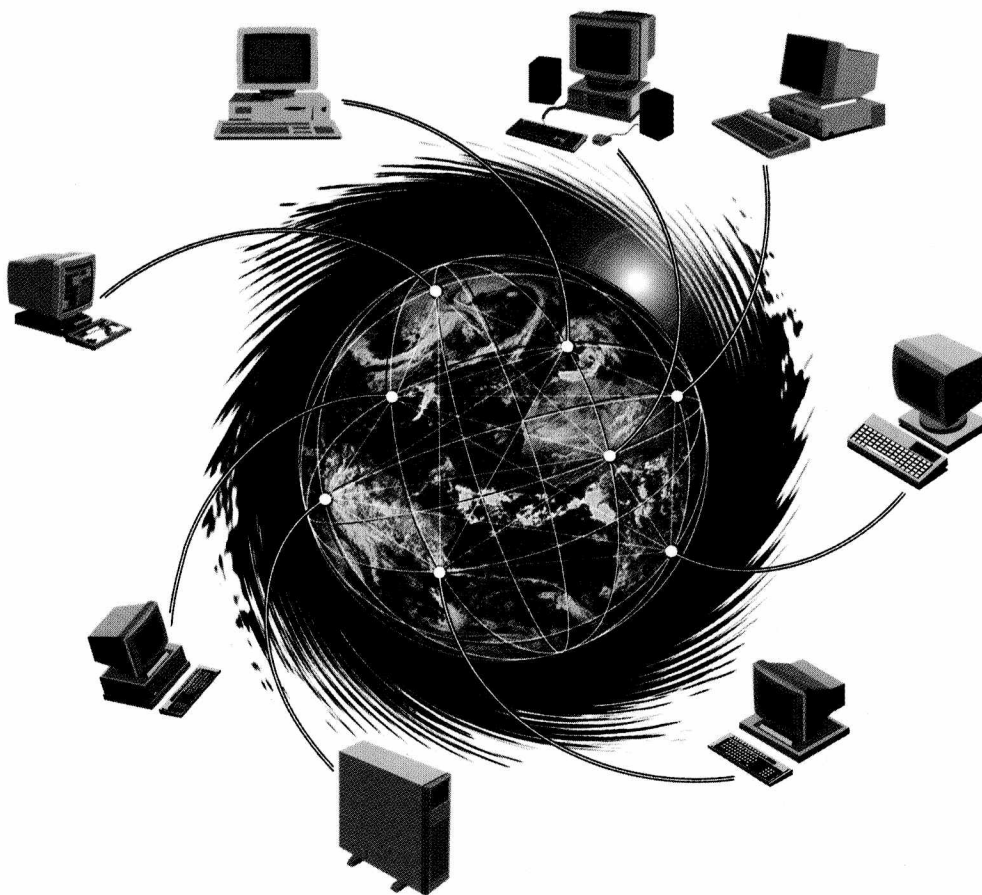
In the 1970s, people began to see the advantage of connecting computers in networks, and the wonders of e-mail and file transfers were born.

In the 1980s, personal computers (PCs) appeared in great numbers. Soon thereafter, local area networks of interconnected PCs became popular. These networks allowed a local group of PCs to communicate and share such resources as disk drives and printers with each other and with large, centralized multiuser computers.

The 1990s have seen an explosion in computer use, and the hundreds of millions of computers now appearing on every desktop and in almost every home can be connected through the Internet (Figure 1-1), a fact known by every hacker and feared by every bank and government installation.

And the common language of all these computers is fast becoming Java.

FIGURE 1-1
Computers are interconnected through the Internet



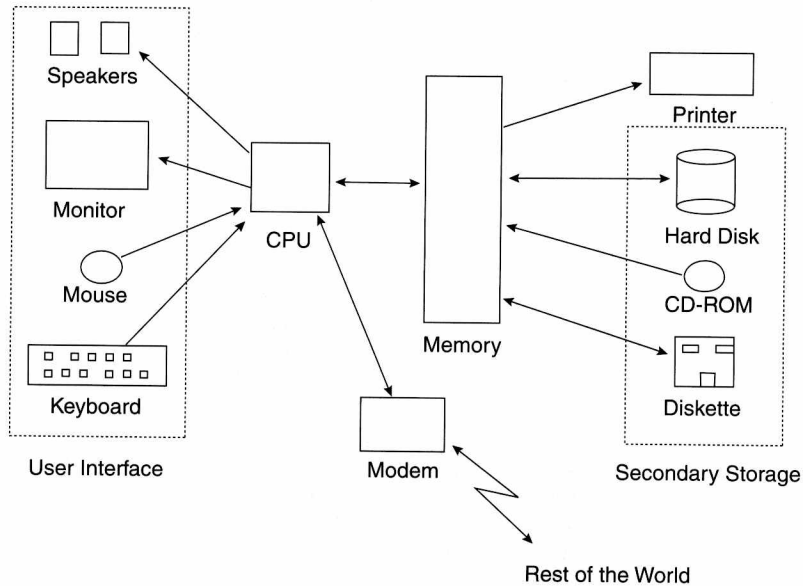
Computer Architecture

Modern computers can be viewed as machines that process information. Information processors consist of two primary components: *hardware* and *software*. Hardware consists of the physical devices that you see on your desktop. Software consists of the programs that enable human beings to use the hardware.

Computer Hardware

A general-purpose computer consists of many interconnected and interacting parts. Figure 1-2 shows the hardware components of a typical PC.

FIGURE 1-2
A typical hardware setup



Input devices send information to the computer for processing. Examples of input devices include:

- A keyboard for entering text.
- A microphone for entering sound.
- A mouse for direct manipulation of images on the monitor screen.
- A *modem* for entering information from other computers.

Output devices display information in a form that people can understand. Examples of output devices include:

- A monitor for displaying text and images on a screen.
- Speakers for emitting sound.
- A printer for producing hard copies of text and images.

Secondary storage devices store information that must be retained on a permanent or semipermanent basis. Examples are disks and CD-ROMs.

A computer uses two devices to process information: *memory* and a *central processing unit* (CPU). The memory (sometimes also called *main memory* or *primary memory*) consists of a large number of cells that can contain information. Each cell is an electronic device that can be in one of two states, either on or off. A given pattern of these states can be used to represent any information whatsoever, such as numbers, text, images, and sound.

Some of the information stored in memory represents *data*, or the information to be processed. The rest of the information stored in memory represents instructions, which tell the computer how to process the data. In other words, both the *program* (the instructions) and the information to be processed (the data) are stored as patterns of electronic states in memory.