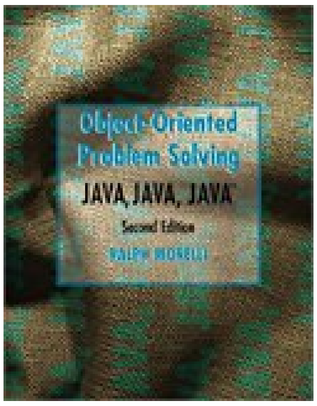
[PDF] Java, Java, Java Object-Oriented Problem Solving (2nd Edition)

Ralph Morelli - pdf download free book



Books Details:
Title: Java, Java, Java Object-Orien
Author: Ralph Morelli
Released: 2002-02-21
Language:
Pages: 862
ISBN: 0130333700
ISBN13: 978-0130333704
ASIN: 0130333700

pdf, mobi, epub, azw, kindle

Description:

From the Inside Flap Preface Who Should Use This Book?

The topics covered and the approach taken in this book are suitable for a typical depthfirst Introduction to Computer Science (CS1) course or for a slightly more advanced Java as a Second Language course. The book is also useful to professional programmers making the transition to Java and object-oriented programming.

The book takes an "objects first" approach to programming and problem solving. It assumes no previous programming experience and requires no prior knowledge of Java or object-oriented programming. Why Start with Objects?

Java, Java, Java takes an "objects early" approach to teaching Java, with the assumption that teaching beginners the "big picture" early gives them more time to master the principles of object-oriented programming.

The first time I taught Java in our CS1 course I followed the same approach I had been taking in teaching C and C++ - namely, start with the basic language features and structured programming concepts and then, somewhere around midterm, introduce object orientation. This approach was familiar, for it was one taken in most of the textbooks then available in both Java and C++.

One problem with this approach was that many students failed to get the big picture. They could understand loops, if-else constructs, and arithmetic expressions, but they had difficulty decomposing a programming problem into a well organized Java program. Also, it seemed that this procedural approach failed to take advantage of the strengths of Java's object orientation. Why teach an object-oriented language if you're going to treat it like C or Pascal?

I was reminded of a similar situation that existed when Pascal was the predominant CS1 language. Back then the main hurdle for beginners was procedural abstraction - learning the basic mechanisms of procedure call and parameter passing and learning how to design programs as a collection procedures. Oh! Pascal!, my favorite introductory text, was typical of a "procedures early" approach. It covered procedures and parameters in Chapter 2, right after covering the assignment and I/O constructs in Chapter 1. It then covered program design and organization in Chapter 3. It didn't get into loops, if-else, and other structured programming concepts until chapter 4 and beyond.

Presently, the main hurdle for beginners is object abstraction. Beginning programmers must be able to see a program as a collection of interacting objects and must learn how to decompose programming problems into well designed objects. Object orientation subsumes both procedural abstraction and structured programming concepts from the Pascal days. Teaching "objects early" takes a top-down approach to these three important concepts. The sooner you begin to introduce objects and classes, the better the chances that students will master the important principles of object orientation.

Object Orientation (OO) is a fundamental problem solving and design concept, not just another language detail that should be relegated to the middle or the end of the book (or course). If OO concepts are introduced late, it is much too easy to skip over them when push comes to shove in the course.

Java is a good language for introducing object orientation. Its object model is better organized than C++. In C++ it is easy to "work around" or completely ignore OO features and treat the language like C. In Java there are good opportunities for motivating the discussion of object orientation. For example, it's almost impossible to discuss applets without discussing inheritance and polymorphism. Thus rather than using contrived examples of 00 concepts, instructors can use some of Java's basic features applets, the class library, GUI components - to motivate these discussions in a natural way.

Key Features

In addition to its objects early approach, this book has several other important features.

The CyberPet Example. Throughout the text a CyberPet class is used as a running example to motivate and illustrate important concepts. The CyberPet is introduced in Chapter 2, as a way of "anthropomorphizing" the basic features of objects. Thus individual CyberPets belong to a class (definition), have a certain state (instance variables), and are capable of certain behaviors like eating and sleeping (instance methods). Method calls are used to command the CyberPets to eat and sleep. In Chapter 3 the emphasis is on defining and using methods and parameters to promote communication with Cyberpets. In subsequent chapters, concepts such as inheritance, randomness, animation, and threads are illustrated in terms of the CyberPet. Some of the lab and programming exercises are also centered around extending the behavior and sophistication of the CyberPet.

Applets and GUIs. Applets and GUIs are first introduced in Chapter 4 and then used throughout the rest of the text. Clearly, applets are a "turn on" for introductory students and can be used as a good motivating factor. Plus, event-driven programming and Graphical User Interfaces (GUIs) are what students ought now to be learning in CS1. We are j long past the days when command-line interfaces were the norm in applications programming. Another nice thing about Java applets is that they are fundamentally object oriented. To understand them fully; students need to understand basic OO concepts. That's why applets are not introduced until Chapter 4, where they provide an excellent way to motivate the discussion of inheritance and polymorphism.

Companion Web Site. The text is designed to be used in conjunction with a companion Web site that includes many useful resources, including the Java code and Java documentation (in HTML) for all the examples in the text, additional lab and programming assignments, on-line quizzes that can be scored automatically, and PowerPoint class notes. Problem Solving Approach. A pedagogical, problem solving approach is taken throughout the text. There are total of 13 fully developed case studies, as well as numerous other examples that illustrate the problem solving process. Marginal notes in the text repeatedly emphasize the basic elements of object-oriented problem solving: What objects do we need? What methods and data do we need? What algorithm should we use? And so on. Self-study Exercises. The book contains more than 200 self-study exercises, with answers provided at the back of each chapter. End-of-Chapter Exercises. Over 400 end-of-chapter exercises are provided, including "Challenge" exercises at the end of most sets. The answers are provided in an Instructor's Manual, which is available to adopters. Programming, Debugging and Design Tips. The book contains nearly 400 separately identified "tips" (Programming Tips, Debugging Tips, Effective Design Principles, and Java Language Rules) that provide useful programming and design information in a nutshell. Laboratory Sections. Each chapter concludes with a laboratory exercise, so the text can easily be used to support lab-based CS1 courses (such as ours). For CS1 courses that are not lab-based, these sections can still be read as preparation for a programming assignment. For each lab in the text, the companion Web site contains additional resources and handouts, as well as a repository of alternative lab assignments. From the Library Sections. Each chapter includes a section that introduces one or more of the library classes from the Java API (Application Programming Interface). In the early chapters these sections provide a way of introducing tools, such as I/O classes and methods, needed to write simple programs. In subsequent chapters, some of these sections introduce useful but optional topics, such as the NumberFormat class used to format numeric output. Others introduce basic GUI (Graphical User Interface) components that are used in program examples and the laboratory sections. Object-Oriented Design Sections. Each chapter includes a section on Object-Oriented Design which is used to underscore and amplify important principles such as inheritance, polymorphism, and information hiding. Java Language Summary. Those chapters that introduce language features contain Java Language Summary sections that summarize the feature's essential syntax and semantics. Organization of the Text

The book is organized into three main parts. The first part (Chapters 0 through 4) introduces the basic concepts of object orientation, including objects, classes, methods, parameter passing, information hiding, inheritance, and polymorphism. Although the primary focus in these chapters is on object orientation, rather than Java language details, each of these chapters has a Java Language Summary section that summarizes the language elements introduced.

In Chapters 1 to 3 students are given the basic building blocks for constructing a Java program from scratch. Although the programs at this stage have limited functionality in terms of control structures and data types, the priority is placed on how objects are constructed and how they interact with each other through method calls and parameter passing. --This text refers to an out of print or unavailable edition of this title.

From the Back Cover

This second edition of *Java, Java, Java* offers a robust, accessible, and flexible problem-solving perspective. The use of Unified Modeling Language (UML) diagrams throughout the text, strongly emphasizes object-oriented design. This book assists students and professionals with their most challenging problem as beginning programmers: object abstraction, or how to use interacting objects and methods.

Using a top-down approach, the author focuses on problem decomposition and program design from the beginning. This methodology□along with its lucid and engaging exercises and analogies□sets this book apart. Morelli introduces advanced Java features including GUI's (e.g., AWT and Swing), exceptions, threads, files, and sockets. The adaptable and accessible style allows instructors to choose which advanced concepts to teach to introductory students, while intermediate-level programmers can benefit from its thorough, advanced feature coverage.

Java, Java, Java's Numerous Distinguishing Innovations:

- Emphasizes early OO design concepts such as inheritance and information hiding.
- Uses UML diagrams throughout to emphasize object-oriented design.
- Features GUI elements and applets to captivate and maintain the reader's interest while introducing real-world examples.
- Incorporates action-learning techniques such as "Hands on Learning" sections, CyberPet examples. and drop-in boxes on effective design, programming and debugging tips, and Java language rules.
- Covers advanced features of Java: GUI's, graphics and drawing; exceptions; recursive problem solving;. threads and concurrent programming; files, streams, and input/output techniques; sockets and networking; and data structures.
- Includes a Companion Website with extensive supplementary resources, such as a Study Guide, PowerPoint slides, and Java code **www.prenhall.com/morelli**

- Title: Java, Java, Java Object-Oriented Problem Solving (2nd Edition)
- Author: Ralph Morelli

- Released: 2002-02-21
- Language:
- Pages: 862
- ISBN: 0130333700
- ISBN13: 978-0130333704
- ASIN: 0130333700