PREFACE

This book teaches the fundamentals of programming by way of the Java language. It assumes no prior programming experience and begins with the basics, such as how to compile and run a Java program. Next, it discusses the keywords, operators, and constructs that form the Java language. The book also covers several parts of the Java Application Programming Interface (API) library, including Swing, which is the framework used to create programs that have a graphical user interface (GUI), and the Collections Framework, which is used to store collections of objects. In short, this book is intended as a comprehensive introduction to Java. Like most computer languages, Java has evolved over time. At the time of this writing, the latest version is Java 7 (JDK 7), and this is the version of Java covered by this book. However, most of the material is also applicable to other recent versions of Java, such as version 6.

A STRAIGHT AHEAD APPROACH

This book uses what we characterize as a "straight ahead" approach. By this, we mean that topics are introduced in a cohesive sequence that is intended to keep the focus of each discussion on the topic at hand. This approach simplifies and streamlines the presentation. On occasions when a departure from the main presentation flow is necessary, we attempt to do so in a way that minimizes the disruption. The goal of our approach is to present the Java language in a way that clearly shows the interrelationship of its parts, rather than as a jumble of disconnected features.

To help manage the material, this book is organized into three parts. Part One describes the elements that define the Java language and the core elements of programming. It begins with an overview of Java followed by the basic concepts of data types, operators, and control statements. It then progressively introduces the more sophisticated features of the language, such as classes, methods, inheritance, interfaces, packages, exceptions, multithreading, and generics. Part One also describes I/O, because it is integral to many Java programs, and applet fundamentals, because the applet is a quintessential Java application. Part One ends with a chapter on object-oriented design.

As it relates directly to Part One, our "straight ahead" approach keeps the focus on the elements of the Java language and the fundamentals of programming, with each new section building on the foundation of what comes before. Where possible, we avoid digressions that distract from the main topic. For example, discussions of GUI programming via Swing are handled in Part Two, rather than being intermixed with discussions of basic concepts. This way, the presentation in Part One remains firmly rooted in the core issues of Java and of programming.

Part Two introduces Swing. It begins with an overview of GUI programming with Swing, including the basic concepts of components, events, and layout managers. Subsequent chapters advance in an orderly fashion, presenting an overview of several Swing components, followed by menus, dialogs, painting, and so on. This "straight ahead" approach is intended to help students more easily integrate each new feature into the overall picture they are forming of the Swing framework.

Part Three explores portions of the Java API library. Because the API library is very large, it is not possible to discuss it in its entirety in this book. Instead, we focus on what we consider to be those parts of the library with which every Java programmer should be familiar. In addition to covering large portions of **java.lang** and **java.util** (with special emphasis on the Collections Framework), we also present an overview of networking, and introduce the concurrency API, including the Fork/Join Framework. The material is presented in a straightforward manner that is designed to give the student a solid overview of several core library elements.

OBJECTS SOON…BUT NOT TOO SOON

One of the first commonly-asked questions about a programming book is whether it uses an "objects early" or an "objects late" approach to teaching the key tenets of object-oriented programming. Of course, what constitutes "early" or "late" can be somewhat subjective, and neither term precisely describes the organization of this book. The phrase we use to characterize our approach is "objects soon, but not too soon." Our goal is to introduce objects at the appropriate time for the student. We believe that this is not until key features of the language have been learned.

Towards this end, the focus of the first three chapters is on the fundamentals of the Java language, such as its syntax, data types, operators, and control statements. We believe that mastery of these elements is a necessary first step because they form the foundation of the language, and the foundation of programming in general. (In other words, it is difficult to write meaningful programs without understanding these elements.) In our view, only after the basic elements of a program have been learned, is the student ready to move forward to objects.

After the book has covered the fundamentals, objects are introduced in Chapter 4, and from that point on, object-oriented features, techniques, and concepts are integrated into the remaining chapters. Additionally, objects are introduced in a carefully paced, step-by-step fashion. This is intended to help the student grasp each new feature in context, and without being overwhelmed.

PEDAGOGICAL FEATURES

This book includes several pedagogical elements to facilitate and reinforce learning. Each feature helps ensure that students are fully aware of key skills, can gauge their advancement, and can verify that all concepts are learned.

- Key Skills & Concepts: Each chapter begins with a list that identifies the key skills and concepts presented in the chapter.
- Ask the Expert: At various points throughout the book are Ask the Expert boxes. These contain additional information or interesting commentary about

a topic, and use a Question/Answer format. They provide supplemental information without disrupting the main presentation flow.

- Try This Elements: Each chapter contains one or more Try This elements. These are step-by-step examples that walk through the development of a program that demonstrates an aspect of Java related to the chapter's topic. Typically, these are longer examples that show a feature in a more practical setting.
- Progress Checks: Throughout each chapter, Progress Checks are presented to test the student's understanding of the preceding section. The answers to these questions are at the bottom of the same page.
- Exercises: Each chapter concludes with exercises that include short answer, fill-in-the-blank, and true/false questions, and coding exercises. The answers to selected exercises are in Appendix C.

ACM RECOMMENDATIONS

The 2008 update to the ACM Curricula Recommendations (http://www.acm.org/education/curricula/ComputerScience2008.pdf) recommends that all computer science students be fluent in at least one programming language and have some understanding of object-oriented and event-driven programming. We believe that students who learn the material covered by this book will have the desired knowledge and skills. We have included in the book not just an introduction to programming using the Java language, but broader coverage that includes advanced Java features, the Swing framework, and large parts of several important API packages.

The first part of the book covers a significant portion of the topics in the Programming Fundamentals (PF) knowledge area of the ACM Recommendations (the main exceptions being the knowledge units FoundationsInformationSecurity and SecureProgramming). The first part also includes a chapter on object-oriented design, which covers a number of the topics in the PL/ObjectOrientedProgramming and SE/SoftwareDesign knowledge units. The second part of the book, which introduces GUI programming with Swing, addresses some of the topics in the knowledge unit HC/GUIProgramming. The third part includes, among others, topics that relate to concurrency. In fact, we devote Chapters 12 and 27 to multithreading and concurrency because we feel, as the ACM Curricular Recommendations discuss, that concurrency is becoming increasingly relevant to the discipline of computer science.