Preface

One of Woody Allen's jokes goes like this:

> I took a speed reading class. I read *War and Peace* in ten minutes. I think it's about Russia.

War and Peace has always been the undisputed exemplar of a book too long to read. However, today it has competition for that honor from all the standard Java™ textbooks. These textbooks have enormous page counts—right up there with *War and Peace*. And they are harder to read. Most students have neither the time nor the diligence to plow through the typical 1,000 page textbook.

Our textbook, *An Introduction to Programming Using Java*, is the solution to the problem. It's not an outline or a reference, but a comprehensive, easy-to-read tutorial. Its most important feature, however, is its size. Its 448 pages cover just about everything the 1,000 page textbooks cover and, *mirabile dictu*, include a lab manual as well.

Of course, a textbook with 448 pages cannot possibly have everything a 1,000 page textbook has. But with Java textbooks, more is less. Back in the days of my youth, when I was in the Army, the master sergeant would occasionally assemble the troops to inform us of the latest news, such as the times at which he would be bringing us bagels and lox. But instead of simply saying "I'll be there at 10 a.m. so you can sleep in until then," he would ramble on endlessly about something or other. I could never pay attention for more than

a minute or so. When our assemblies ended, I always had two reactions: "Boy, I'm glad that's over," and "What did he say?" The fact is, the more he said, the less I absorbed. The same is true with Java textbooks. The longer they are, the less a student learns from them.

We achieved the compression of 1,000 pages of material into 448 pages by trimming four areas in which excessive verbiage is typically found:

- **Overviews**
  Overviews are important but they should be brief and, most important, intelligible to the student. Some Java textbooks start with an extensive discussion of object-oriented programming—encapsulation, information hiding, inheritance, and polymorphism—much of which is incomprehensible to a beginning student. We do discuss object-oriented programming in Chapter 1, but we limit our discussion to a level that a beginner can comprehend.
- **Programming examples**
  Students learn the most by writing programs—not by reading programs. Programming examples should be as simple as possible. Simple concepts should not be illustrated with complex programs.
- **Warnings of potential errors**
  Some textbooks warn students of the errors they are likely to make. A better approach is to have students intentionally make errors in the lab through guided exercises. Students learn and retain more by making mistakes than by reading about making mistakes.
- **Sidebars**
  Sidebars are distracting. Reading a text with many sidebars is like attending a class in which two professors are lecturing simultaneously. Sidebars presumably contain optional material. But, perhaps, some sidebars are not really optional. Thus, for every sidebar, the reader has the distraction of deciding whether to read it.

One unusual feature of our book is the inclusion of sample exams at various checkpoints in the text. The idea of sample exams came from a professor, now retired, who many years ago made the following observation about his classes: If he covers topics A, B, C, and D, and tells students before an exam that they are responsible for topics A, B, C, and D, they study A, B, C, D, and do well on the exam. However, if he simply says that they are responsible for the topics covered, students inexplicably won't study one or more of the required topics. In this case, students do poorly on the exam. "Oh," a student would say. "I didn't know we were responsible for topic C." And the professor would reply, "But we covered topic C two weeks ago. How could you not know you were responsible for it?"

Students respond to specificity. So before every exam, I give a sample exam as a homework assignment. If I want my students to know A, B, C, and D, I give a sample exam on

A, B, C, and D. And as sure as the sun rises in the east, my students learn A, B, C, and D before the real exam. The sample exam is an *unbelievably* effective motivator. Try it!

Other noteworthy features of the book:

- Introduces objects early (in Chapter 1).
- Provides plenty of material for a one-semester course.
- Uses the computer in the lab exercises to teach students some of the fine points of Java. This approach is more efficient, effective, and stimulating than a tedious enumeration in the text proper.
- Each lab includes a set of prep questions. Requiring students to hand in the answers to these questions at the *beginning* of a lab will ensure that they are adequately prepared to do the lab.
- Each lab provides enough exercises to keep even the speediest student working productively during a two or three hour lab session.
- Introduces linked structures. I have found that introducing linked structures in the first programming course makes for a more successful second course (on data structures).
- Explains abstract classes and interfaces in the context of generic programming. With this approach, students quickly grasp the conceptual and technical aspects of these constructs.
- Liked by students. It gives them precisely what they need from a textbook.
- Available supplements include source code for all the programs in the textbook, PowerPoint® and PDF slides, and an Instructor's Manual.

The introductory computer course is an important component in any math, science, or engineering curriculum. Don't use a textbook that will make students say, "I think it's about Java." We believe we have a better alternative.

Laura and I would like the thank Professor Andy Pletch for his helpful critique, Tim Anderson at Jones & Bartlett Learning for his assistance with this project, and Mom for everything else.

Anthony J. Dos Reis
Laura L. Dos Reis