Java Au Naturel

Guide to Object Oriented Design

Java 2 with Swing

Dr. William C. Jones, Jr.
email jonesw@ccsu.edu
Department of Computer Science
Central Connecticut State University
web page: www.javabook.org
Fourth Edition published 5/26/04

Java Au Naturel
Table Of Contents

Chapter 7 Arrays 41 pages

Chapter 8 Elementary Graphics 40 pages

Chapter 9 Exception-Handling 38 pages

Chapter 10 Event-Driven Programming 44 pages

Chapter 14 Stacks, Queues, and Linked Lists 42 pages

Chapter 15 Collections and Linked Lists 40 pages

Chapter 16 Maps and Linked Lists 42 pages

Chapter 17 Binary Trees 42 pages

total 848 pages = 10 + 352 (ch1..9) + 234 (ch10..15) + 196 (ch16..20) + 56 (appendix)

Preface for Students

Software development with an object-oriented approach is the fundamental subject of this book. Java is a programming language used to create the animations you see when you browse the web. Of all the programming languages whose use is wide-spread, Java is the best for learning and doing object-oriented software development. That is why Java is used in this computer science book.

You only learn by doing. So in this book, you will work with many different situations where software development is necessary. That way, you can see how to apply newly-learned techniques in a variety of contexts. When you need a language feature to accomplish a certain purpose, that is when you learn it. If it has several alternatives and you do not need them at that point or in the near future, you do not learn them then; you wait until you need them.

When you learn to speak a natural language such as German or Spanish, you start with a few sentences that are useful in key situations. You gradually expand your repertoire of sentences. You learn grammatical principles that apply to several sentences you already know how to use.

You do not start by learning all the conjugations of verbs and all the declensions of nouns. Incremental development is far more effective: Add items a few at a time, mastering those before you add more. And, most important, learn what you can immediately use in realistic conversations.

You should learn a programming language the same way. You should not start by learning Java's eight different primitive types or its five different control statements. You should start by seeing how to develop software for a particular realistic situation. And you should learn the language features you need for the situation as the need arises.

This book starts by introducing you to objects that make drawings. In this context you learn to send messages to objects to carry out simple tasks. More important, you learn how to "teach" those objects to put together a sequence of simple actions to perform a complex task. These "better-trained" objects have new capabilities in addition to the ones they inherit from the original specifications. This **inheritance** technique of course makes your job much easier. You can do bigger jobs with less work and less chance of getting it wrong if you have objects take over most of the work. When objects are used this way, they should be thought of as your agents or executive assistants.

Chapter One explains the details of compiling and executing a program. It introduces much of the vocabulary you need. It establishes the framework for Java programs. And it gives an overview of the book with a look at other contexts where objects are used to perform useful tasks. This is a foretaste of what is to come, so you are not expected to fully understand at this point everything it mentions.

Chapter Two introduces a quite different software context. The software provided to you defines objects that control the basic physical actions of electronic equipment. Your job is to develop additional software that puts these basic actions together in combinations that perform tasks that the purchaser of the electronic equipment finds useful. You learn to use inheritance in this new context. Then you see how to have objects select between two courses of action depending on the circumstances. You also learn a simple method of diagramming the relationships among classes of objects. It is a widely used technique that is part of the Unified Modeling Language (UML). The UML is the industry standard for modeling software.

Chapter Three shows you how to teach objects to repeat a sequence of actions many times until a task is accomplished. This ability leads to more complex programs, so we discuss a reliable process for developing the logic to solve a problem and translate it into a Java program. This is the one chapter that does not introduce a new major context for developing software.

Chapter Four switches to the context of game-playing programs. In this context, you learn how to build objects "from scratch", specifying what they know as well as what they can do. By this time you can write interesting and useful programs using only the standard library of objects that comes with every Java installation. The game-playing software interacts with the outside world through the keyboard and screen rather than via signals to and from electronic equipment or to a drawing surface.

Chapter Five completes the presentation of basic language features you need for working with objects in your programs. At this point you can construct a complete string-based simulation of the electronic equipment you worked with in earlier chapters. A case study on networks lets you see the interaction of all of the object-oriented concepts and most of the language features applied in another context. The situation that the networking material describes is key to many important real-life problems. The analysis and solution of some of these problems can be quite complex; it is the subject of more advanced courses in computer science. But the discussion here is quite elementary.

Chapter Six expands your arsenal of basic types of values to include characters and numbers with decimal points (heretofore you only had the whole-number and true-false kinds of values). It also gives you a full set of methods for dealing with strings of characters. These language features are introduced in the context of software to schedule work orders at a car repair shop using the Model/View/Controller approach and several kinds of objects at once.

Chapter Seven gives you the tools you need to work with large masses of data. This is in the context of software to handle a database of people working for a particular company. A re-implementation of a simulation of the electronic-equipment software from Chapters Two and Three helps you solidify your understanding of the key concept of arrays.

The first seven chapters present the features of Java most frequently used in this book, together with a moderate number of examples. Many of the concepts, especially the various uses of arrays, cannot be learned well enough without a great deal of practice. The remaining eleven chapters give you that practice. Their primary purpose is to (a) improve your understanding of principles and techniques of software design and development, and (b) reinforce the concepts in the basic first seven chapters.

Each of the next twelve chapters presents a different software design and development situation. The emphasis is on techniques for creating quality software.

In summary

- Chapter One: Turtle objects can do simple tasks such as draw a line or move to another position. You use Java to combine these simple actions into complex tasks such as having a Turtle draw a flower garden.
- Chapters Two and Three: Stick figures called Vic objects can do simple tasks such as put a compact disk in or out of the current slot and move to the slot before or after the current slot. You use Java to combine these simple actions into complex tasks that move several CDs where you want them.
- Chapters Four and Five: You use Java to define the simple tasks that objects can do. You apply these language skills to define how game-playing objects interact with a human player, and to define how the stick figure objects perform their simple tasks. You may also, if you wish, skip directly to Chapter Eight on graphics to learn to define how Turtle objects perform their simple tasks.
- Chapter Six: You use Java to develop commercial software that defines how several classes of objects perform simple tasks and then has these several classes of objects cooperate to accomplish a complex task.
- Chapter Seven: You use Java for software handling many objects at once.

Re-read this summary once or twice as you progress through the textbook, whenever you get the feeling that you cannot see the forest for the trees.

You cannot avoid heavy technical vocabulary in a first course in computer science. But as a student you have a right to short paragraphs and simple sentences to explain the vocabulary and the difficult concepts accurately. This book makes a special effort to give you just that.

You should work out most of the unstarred exercises in the book. Many of the ones marked "harder" might take you 10 or 15 minutes, so you could leave out some of those. But the others should only take two minutes. The answers to all unstarred exercises are at the end of the corresponding chapter. If you do not have a good answer after five minutes, either continue working until you do or at least study the given answer until you see why it is a good answer and where you went wrong. This will help you gain a thorough understanding of the concepts rather than a superficial one.

This book does not require any knowledge of programming or any mathematics beyond elementary algebra and (in a few places) a bit of trig. But it does require that you practice what you read about. Reading alone will not suffice -- you only learn by doing!