Programming Language Processors in Java: Compilers and Interpreters

Aims to exemplify good software engineering principles at the same time as explaining the specific techniques needed to build compilers and interpreters. Examples included. DLC: Java (Computer program language)

My Personal Review:
This is really the greatest Compiler for Dummies book as of today. I own and studied all books (total arround 80) about compiler constructions you can buy in the world today plus a lot of out of print titles, so I know what you can buy on the market...

If youre looking for a learning-by-coding compiler book dont look futher. This is a great introbook for a Bachelor of Science of a Professional Bachelor student! If you have the money to buy only one compiler book buy this one. If you have the money to buy 2 books and you also know a bit C++, buy this one and Writing compilers and Interpreters from Ronald Mak which also offers a pratical approach which is also nice for newbies but not that nice as this book.

If you have finished this book and want to get more theoretical insight or youre a Computer Science M.Sc student (like me), read Programming Language Pragmatics from Mr. Scott, which dont present the material on a dry manner. This book covers all aspects of compilation and language design in greater detail! Be sure you have read (or master) an intro text like programming language processors in java before starting books like programming language pragmatics.

If youre a M.Sc. student and youre looking for a learning-by-coding book i recommend the books of Andrew Appel (Modern Compiler Implementation in Java) which covers advanced topics (optimization, register selection,etc). But beware: if youre new to compiler design forget Andrew
Appels book, and buy this one because Appels would be a bit too difficult for you.

Back to the Programming language processors in Java book.

This is what I like and what not:

PROS:

* It gives some nice written theoretical introduction of the whole compilation process (e.g. what's LL and LR-parsing, how is runtime organization organized (stack, parameter passing, stackframes etc). It doesn't dive too deep, but you will be familiar with the topics. E.g. it explains how LL and LR parsing works (with some nice examples how to parse LL/LR simple English sentences), but it doesn't tell you what the advantages/disadvantages of both methods are. This is beyond the scope of this book. For these topics read later something else (e.g. programming language pragmatics).

* Not all Java code is printed to fill 1000 pages. Every codesnippet is well commented, all repeatative code is left out: you can download all the Java code. So this book is not one big listing.

* It gives you insight how to build a really nice Virtual Machine. The author is talking about an interpreter, but the compiler generates modern intermediate code (STORE, LOAD, CALL, JMP) and the VM execute this in a big WHILE-loop.

* Some learning-by-coding compiler books (e.g. Building your own compiler with C++ by Jim Holmes (not recommended)) explain only a silly subset of pascal (e.g. only assignment and writeln), but this book teach you mini Triangle-language which also offers arrays, functions/procedures, records (structs), and parameter passing by reference/value. It also explains you those more difficult topics like parameter by reference passing is handled by the codegenerator.

* It has a nice chapter about runtime organization. This is a nice chapter for people which are new to e.g. processor architecture. This chapter explains you how local variables are stores (stack), how parameter passing to functions is working and how return values are passed back. Because the Interpreter (VM) which is introduced in the book, has a modern stack machine VM (STORE, LOAD, JMP) this is an excellent way to study those VMs and code generating for a stack machine.

CONS:

* I agree to the customer review of Mr. Yegge of July 12, 2004 that the Java Code is not always supernice. E.g. there is often java.lang.Object parameter passing which is later dangerous narrative casted. I do NOT
agree with Mr. Yegge about his remarks on the Visitor pattern. The author explains why he is using the visitor pattern: to reduce coupling between the CodeGenerator or TypeChecker and the AST. On this point I like the design of the author. The idea of using the visitor pattern is nice, but it is somewhat bad implemented with those narrative casts.

Conclusion: I STRONGLY recommend this book for people who are new to compiler design.

For More 5 Star Customer Reviews and Lowest Price:
Programming Language Processors in Java: Compilers and Interpreters - 5 Star Customer Reviews and Lowest Price!