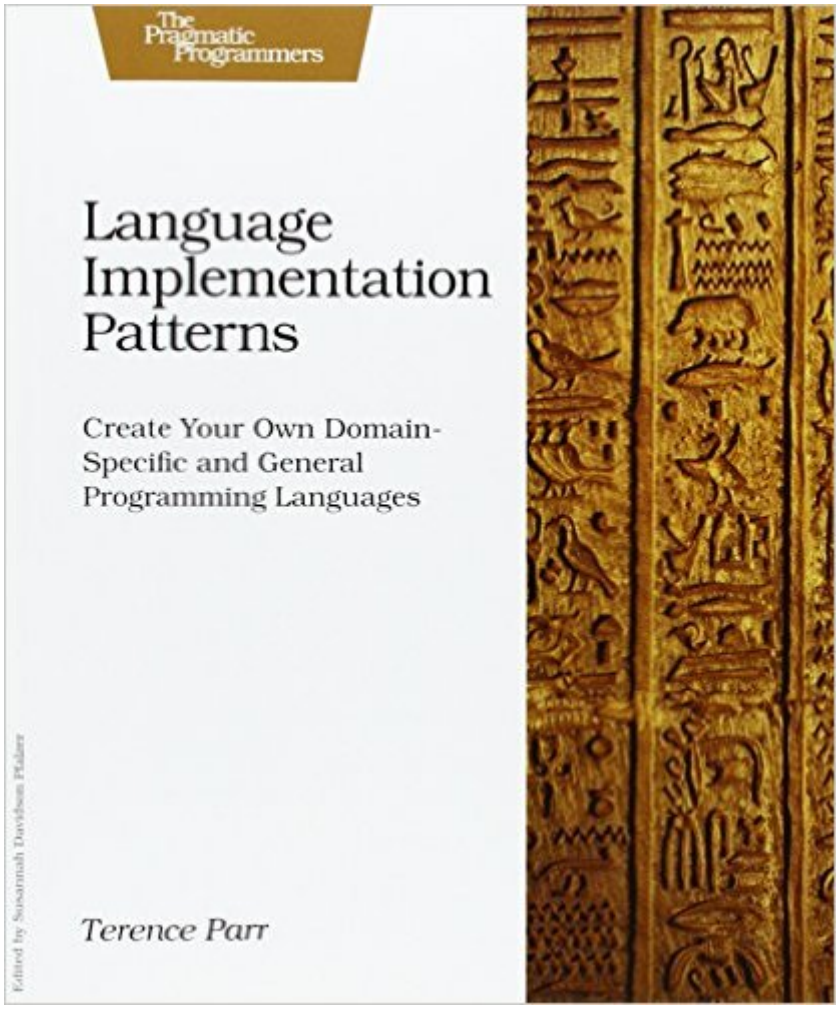
Language Implementation Patterns: Create Your Own Domain-Specific And General Programming Languages (Pragmatic Programmers)



Synopsis

Learn to build configuration file readers, data readers, model-driven code generators, source-to-source translators, source analyzers, and interpreters. You don’t need a background in computer science--ANTLR creator Terence Parr demystifies language implementation by breaking it down into the most common design patterns. Pattern by pattern, you’ll learn the key skills you need to implement your own computer languages.Knowing how to create domain-specific languages (DSLs) can give you a huge productivity boost. Instead of writing code in a general-purpose programming language, you can first build a custom language tailored to make you efficient in a particular domain.The key is understanding the common patterns found across language implementations. Language Design Patterns identifies and condenses the most common design patterns, providing sample implementations of each.The pattern implementations use Java, but the patterns themselves are completely general. Some of the implementations use the well-known ANTLR parser generator, so readers will find this book an excellent source of ANTLR examples as well. But this book will benefit anyone interested in implementing languages, regardless of their tool of choice. Other language implementation books focus on compilers, which you rarely need in your daily life. Instead, Language Design Patterns shows you patterns you can use for all kinds of language applications.You’ll learn to create configuration file readers, data readers, model-driven code generators, source-to-source translators, source analyzers, and interpreters. Each chapter groups related design patterns and, in each pattern, you’ll get hands-on experience by building a complete sample implementation. By the time you finish the book, you’ll know how to solve most common language implementation problems.

Book Information

Series: Pragmatic Programmers

Paperback: 374 pages

Publisher: Pragmatic Bookshelf; 1 edition (January 10, 2010)

Language: English

ISBN-10: 193435645X

ISBN-13: 978-1934356456

Product Dimensions: 7.5 x 0.8 x 9.2 inches

Shipping Weight: 1.4 pounds (View shipping rates and policies)

Average Customer Review: 4.5 out of 5 starsÂ Â See all reviewsÂ (30 customer reviews)

Best Sellers Rank: #173,829 in Books (See Top 100 in Books) #10 inÂ Books > Computers &

Technology > Programming > Languages & Tools > Compiler Design #24 inÂ Books > Computers & Technology > Programming > Languages & Tools > Compilers #41 inÂ Books > Computers & Technology > Programming > Microsoft Programming > .NET

Customer Reviews

I was terribly interested in getting my hands on this book since I'm taking a formal course on Compilers and Interpreters at university and I really wanted to know: What's the difference between what we (as computer scientists) are taught in a compilers' course and the more practical approach presented in the book?As it turns out there's a big difference. If you want to be the ultimate guru of compilers (eg. contributing an even more efficient compiling technique for language X or creating a language that forces us all to reconsider what we know about compilers) you need both, the theory the practice (because without the theory you wouldn't know how to improve or make obsolete an existing technique, and without the practice you wouldn't be able to put that knowledge to work inside a language compiler). Now if you just want to be able to deal with your DSL (domain specific language), create data readers, code generators, source-to-source translators, source analyzers, etc. you'll love the hands on information presented in this book.Let's be honest, how many of us developers are required or willing to create a language from scratch together with its compiler or interpreter versus the ones that just need to parse an XML file, process a DSL or create a configuration file reader? I would say that there are much more developers in the later group. But fortunately we all (or almost all) share one thing in common: we know software patterns!

This is my favorite kind of book: the harder I work, the more I get. So be prepared to work hard. And if you do, you will be rewarded with gems of insight. If you're looking for a "cookbook" I respectfully suggest that you examine your reasons for being interested in language design. All the ingredients and kitchen implements are here, clearly labeled and explained. There are even examples of how you might consider mixing them. But it's up to you to write the recipe you need, and a key objective of this book is getting you to that magical moment when you see how everything comes together.Plenty of people much more educated and experienced in the art and science of language design than I am will surely write insightful reviews about the merits of this book from the perspective of specialists. I'm writing this review for the rest of us.Terence Parr continues his campaign to make superb language-development tools accessible. Have you ever wondered how your compiler really works? Maybe you've dreamed about creating your own scripting language -- the one that works the way *you* want -- but you're not Larry Wall. Well, take heart. Professor Parr's

second ANTLR book is here. Maybe you never took a course in compiler design (I haven't.) or maybe you have and are still wondering how to do anything practical with it. This book is for you. You very well might not become the next Guido van Rossum, but you will come away with a deeper appreciation of language implementation -- probably enough to create your personal dream language.I call this the "second ANTLR book," but that's a gross oversimplification.

In high school I created ACID1, an interpreter for the BASIC programming language. ACID1 was never completed but I felt I could take the world down with my very own programming language. That did not happen. ACID2 was born some years later in a college dorm. It did not work either.These failures taught me an important lesson. ACID1 was created with no prior knowledge of language building whatsoever. ACID2 was, however, created with a surplus of theories (the Dragon Book, anyone?). I needed a middle ground.I found Language Programming Patterns by Terence Parr about a month ago. LIP, if you may, since we are quite pleased with K&R, CLRS, TAOCP, etc... When I picked up LIP, my internal alarm went off, I felt it was going to be a good read.Guido van Rossum, creator of Python (Python forever!), commented: Throw away your compiler theory book! So I knew. I also found out that professor Parr has been teaching language applications programming for years. Then I knew. The book itself came from the famous Pragmatic Bookshelf. And I knew: LIP would be a good read.The book did not disappoint me, and will not disappoint any programmer with interests in language applications. LIP is the perfect mix of theory and practice. LIP is the working ACID I did not write.Parr uses Java for his examples. I confess that I barely know Java. But half the patterns I have translated to Python while reading. This wouldn't have been possible without the great explanations in the book.Each pattern has a Purpose, a Discussion, an Implementation, and Related Pattern section. Patterns are grouped together in chapters in such a way that when you've completed the chapter, you have a complete skill.

Professional Made Easy & Android Programming in a Day (C++ Programming, C++ Language, C++for beginners, C++, Programming ... Programming, Android, C, C Programming) Crochet: Easy Crochet Patterns: Crochet Patterns for Beginners (Crochet: Step by Step Crochet, Crochet Patterns, Easy Crochet Patterns, Crochet Patterns for Beginners, and Crochet Projects) Body Language: Body Language Training - Attract Women & Command Respect, by Mastering Your High Status Body Language (Body Language Attraction, Body Language ... Language Secrets, Nonverbal Communication) Debug It!: Find, Repair, and Prevent Bugs in Your Code (Pragmatic Programmers) C#: Programming Success in a Day: Beginners guide to fast, easy and efficient learning of C# programming (C#, C# Programming, C++ Programming, C++, C, C Programming, C# Language, C# Guide, C# Coding) R Programming: Learn R Programming In A DAY! - The Ultimate Crash Course to Learning the Basics of R Programming Language In No Time (R, R Programming, ... Course, R Programming Development Book 1) C#: Design Patterns: The Easy Way Standard Solutions for Everyday Programming Problems; Great for: Game Programming, System Administration, App Programming, ... & Database Systems (Design Patterns Series) Crochet Mandala For Beginners Learn To Create 15 Amazing Crochet Mandala Patterns: (Crochet Mandala Patterns, Crochet for Beginners) (crochet books patterns, cute and easy crochet) Release It!: Design and Deploy Production-Ready Software (Pragmatic Programmers) Good Math: A Geek's Guide to the Beauty of Numbers, Logic, and Computation (Pragmatic Programmers) Agile in a Flash: Speed-Learning Agile Software Development (Pragmatic Programmers) Test Driven Development for Embedded C (Pragmatic Programmers) OpenGL ES 2 for Android: A Quick-Start Guide (Pragmatic Programmers) Practical Vim: Edit Text at the Speed of Thought (Pragmatic Programmers) Ace General Chemistry I and II (The EASY Guide to Ace General Chemistry I and II): General Chemistry Study Guide, General Chemistry Review Ace General Chemistry I: The EASY Guide to Ace General Chemistry I: (General Chemistry Study Guide, General Chemistry Review)