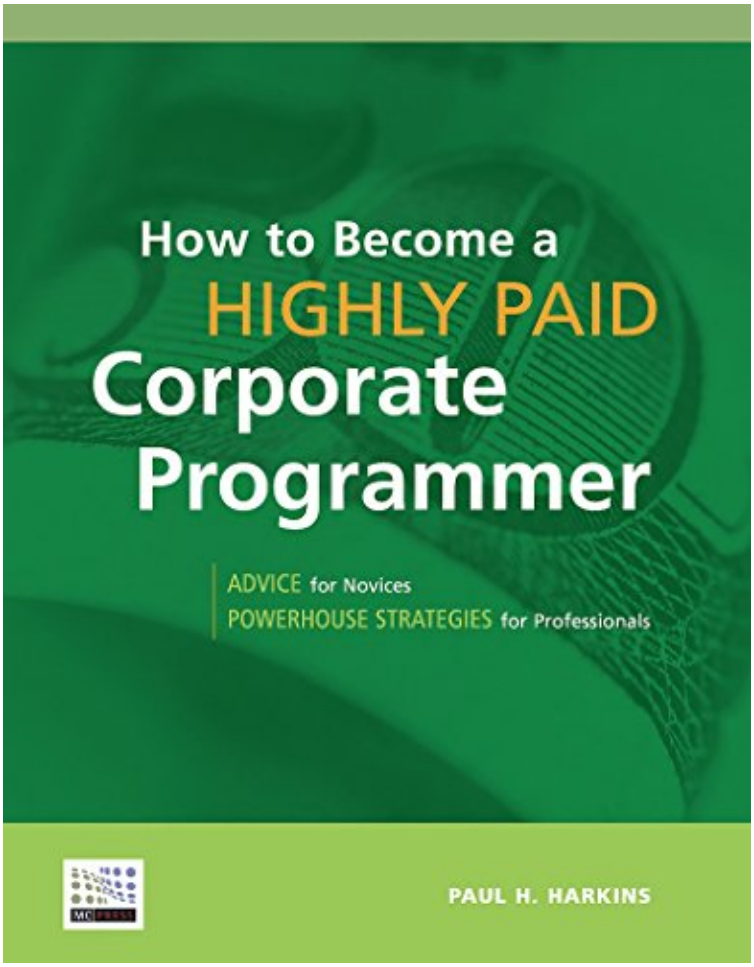
How to Become a Highly Paid Corporate Programmer

Paul H. Harkins
**Download PDF | ePub | DOC | audiobook | ebooks*



#3284419 in eBooks 2004-03-01 2016-02-22File Name: B01C27FE86 | File size: 74.Mb

Paul H. Harkins : How to Become a Highly Paid Corporate Programmer before purchasing it in order to gage whether or not it would be worth my time, and all praised How to Become a Highly Paid Corporate Programmer:

0 of 0 people found the following review helpful. This is a must read for programmers!By Spotlight ReviewsMy copy only cost 1 cent. I've already got real benefits from this book. It's not a new one, but the concepts work.If you want to get ahead in this field, grab a copy. Not a long read, and most of the good stuff is in the 1st 1/3 of the book. I'd be happy to pay for an updated version.0 of 0 people found the following review helpful. Not for OOP, Agile, or any other modern programmer!By avgvstvsThis book might only work if you plan on becoming a Systems/370 or z/OS programmer. I still wouldn't recommend it.Harkins gives advice that no doubt worked for him--but as another reviewerstated, this advice is largely dated or downright personal. For example,early in the book he warns against listening to music while you work,because it bothers "other people." Clearly, this means him. I've foundthat by first, asking my manager on their policy, and then checking with myteammates, this isn't an issue.Now, the reality is that Agile development is picking up alot of steam, andas offices begin rearranging their work areas to account for the "new"process that Agile represents, its highly doubtful that you'll get tolisten to music anyway, but I'm not alone in

noticing that I work well notin silence, but not in total noise.Chapter 8 contains a slew of advice that is simply--avoided at all costs.First and foremost Harkins neatly abandon's OOP. He writes,"The programmer should not split one logical program of a business functioninto 50 parts, calling the 50 sub-programs from the main program and thenhaving the called programs call more programs down many levels (in the callstack). If the business logic, such as validating a customer number, istruly unique to the program, then put it in the program, not in asub-program. "Don't get me wrong, he still suggests separating business logic from dataaccess and ostensibly--presentation. But he flat out states to put allbusiness logic into a single program. For those of you that don't havetheir alarm bells ringing--this destroys code reuse. Someone else in yourcompany can use the solution to your problem (or has already solved it) butif its nested within the logic of a driver/main program, guess who willnever get to see it? My biggest criticism is that if you follow theprocess of putting ALL business logic into your "main" program of somekind, it's not going to be easy to test--at all. And the foundation oftrusting your code is built upon having extensively tested what you can atevery feasible level in your software.I'm more of a follower of Uncle Bob, and share his opinion that if you needto extensively document your code, it's probably poorly written. As muchas possible, code should be self-documenting. Harkins suggests the exactopposite here.Finally, he gives the "sage advice" of testing your program, and notdeleting your production code!!! Really? In the world of coderepositories with robust versioning systems, THIS is news? Even when thiswas written in 2004, CVS was common. And what about CAT?Much of the rest of the advice in the book follows these processes. Theguy has clearly never worked on a complex multi tiered application--most ofthe advice that he gives is salient (maybe) for batch and utility programs. And then hestarts talking about the "virtues" of GOTO...Harkins has written a highly opinionated very narrow book. In short, getthe job done as fast as possible.If you REALLY want to learn "How toBecome a Highly Paid Corporate Programmer," read "Clean Coding," and"Working Effectively with Legacy Code." These two books alone I credit forgiving me a top-paying job out of school.0 of 0 people found the following review helpful. Excellent bookBy MarcusThis book is excellent if you use common sense.Here's the deal, Harkins is an old fox in the mainframe game, so a lot of his examples are from the haydays of mainframe programming.But here's where it seems other reviewers are not paying attention; Everything can be applied to a modern day programmer!Take what's being taught and don't focus so much on the exact programming languages he's using in the examples, and you'll do just fine.That said, people like to forget that a lot of heavy-use critical systems *still run on old mainframes*! And there's a serious shortage of mainframe programmers.The world does not revolve around just the web, you think PHP is what powers a warehouses' computer systems? You think JavaScript handles big-bank number crunching?What Harkins teach is valid, for example: Always work at improving your skill, learn from the best programmers in your department/business, drop skills that will hamper you, add skills that are wanted in the up-and-coming areas of your work, and so on.

Written for those gifted coders and programmers who get wrapped up in how complex their code is and forget the basics of how to be successful in a corporate culture, this guide will help put these programmers on the fast track to promotions and raises. Programmers fresh out of technical school are shown how to make the transition into a corporate setting by managing the interview process, learning from the programmers already in the workplace, and learning the business side of programming. Seasoned programmers are provided with tips on how to please supervisors, how to maximize their worth in corporations, and how to develop mentoring relationships. Advice on moving out of the corporate world and becoming an entrepreneur either in the consulting arena, in writing package software, or in founding a software development company is also provided.

About the AuthorPaul H. Harkins is the president and chief technology officer of Harkins Audit Software, Inc. and is an active corporate programmer. He was a senior systems engineer at IBM for 21 years and a principal in Apparel Business Systems, Inc., a software development firm, where he was responsible for product development and international support. He lives in West Chester, Pennsylvania.