Extracted from:

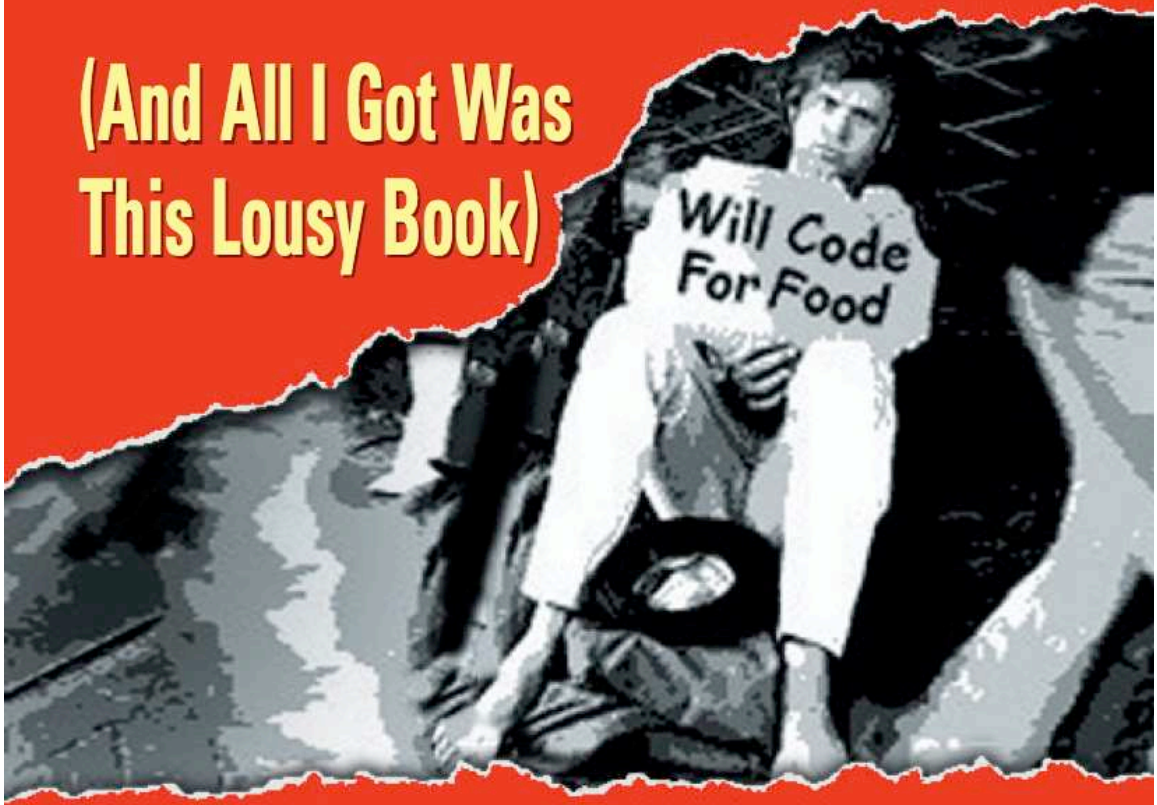
My Job Went to India

And All I Got Was This Lousy Book

The Pragmatic Programmers

MY JOB WENT TO INDIA

(And All I Got Was This Lousy Book)



52 Ways To Save Your Job

Chad Fowler

6 Be a Specialist

How would you write a program, in pure Java, that would make the Java Virtual Machine crash? Dead silence. *Hello?*

I'm sorry. I'm not getting you. Could you repeat the question, please? The voice sounded desperate. I knew from experience that repeating the question wasn't going to help. So, I repeated the question, slowly and more loudly. *How would you write a program, in pure Java, that would cause the Java Virtual Machine to crash?*

Uh...I'm sorry. I've never done this before.

I'm sure you haven't. How about this question: how would you write a program that would NOT cause the JVM to crash? I was looking for really good Java programmers. To start the interview, I asked this person (and all the others I had interviewed that week) to rate himself on a scale of one to ten. He said *nine*. I'm expecting a star here. *If this guy rates himself so high, why can't he think of a single abusive programming trick that would cause a JVM to crash?*

Lack of technical depth.

This was someone who claimed to specialize in Java. If you met him at a party and asked what he did for a living, he would say, "I am a Java developer." Yet, he couldn't answer this simple question. He couldn't even come up with a *wrong* answer. Over two-and-a-half intense weeks of interviewing in Hyderabad, Bangalore, and Chennai, this was the rule—not the exception. Thousands of Java specialists had applied for open positions, nearly none of whom could explain how a Java class loader works or give a high-level overview of how memory management is typically handled by a Java Virtual Machine.

Too many of us seem to believe that specializing in something simply means not knowing about other things.

Granted, you don't have to know these things to hack out basic code under the supervision of others. But, these were supposed to be *experts*.

Too many of us seem to believe that specializing in something simply means you don't know about other things. I could, for example, call my mother a Windows specialist, because she has never used Linux or OS X. Or, I could say that my relatives out in the countryside in Arkansas are country music specialists, because they've never heard anything else.

Imagine you visit your family doctor, complaining about a strange lump under the skin of your right arm. Your doctor refers you to a specialist

to have a biopsy performed. What if that specialist was a person whose only credentials as a specialist were that they didn't attend any classes in medical school or have any experience in residencies that weren't *directly* relevant to the act of performing the specific procedure that they were going to perform on you today? I don't mean that they went *deeper* into the topics related to today's procedure. What if they had just skimmed the surface of these topics, but they didn't know anything else? *What if that machine over there starts beeping during the operation?* you might ask. *Oh, that's never happened before. It won't happen this time. I don't know what that machine does, but it never beeps.*

Thankfully, *most* software developers aren't responsible for life or death situations. If they mess up, it typically results in project overruns or production bugs that simply cost their employers money—not lives.

Unfortunately, the software industry has churned out a whole lot of these shallow specialists, who use the term *specialist* as an excuse for knowing only one thing. In the medical industry, a specialist is someone with a *deep* understanding of some specific area of the field. Doctors refer their patients to specialists, because in certain specific circumstances, the specialist can give them better care than a general practioner.

So, what should a specialist be in the software field? I can tell you what I was searching for in every nook and cranny in South India. I was searching for people who deeply understood the Java programming and deployment environment. I wanted folks who could say "been there—done that" in 80% of the situations we might encounter and whose depth of knowledge could make the remaining 20% more livable. I wanted someone who, when dealing with high-level abstractions, would understand the low-level details of what went into the implementation of those abstractions. I wanted someone who could solve any deployment issue we might encounter or would at least know who to call for help if they couldn't.

This is the kind of specialist who will survive in the changing computer industry. If you're a .NET specialist, it's not just an excuse for not knowing anything *except* .NET. It means that if it has to do with .NET, you are the authority. IIS servers hanging and needing to be rebooted? *No problem.* Source control integration with Visual Studio .NET? *I'll show you how.* Customers threatening to pull the plug because of obscure performance issues? *Give me thirty minutes.*

If this isn't what *specialist* means to you, then I hope you don't claim to be one.

Act on it!

1. Do you use a programming language that compiles and runs on a virtual machine? If so, take some time to learn about the internals of how your VM works. For Java, .NET, and Smalltalk, many books and websites are devoted to the topic. It's easier to learn about than you think.

   Whether your language relies on a VM or not, take some time to study just what happens when you compile a source file. How does the code you type go from being text that you can read to instructions that a computer can execute? What would it mean to write your own compiler?

   When you import or use external libraries, where do they come from? What does it *actually* mean to import an external library? How does your compiler, operating system, or virtual machine link multiple pieces of code together to form a coherent system?

   Learning these facts will take you several steps closer to being an expert specialist in your technology of choice.

2. Find an opportunity—at work or outside—to teach a class on some aspect of a technology that you would like to develop some depth in. As you'll see in *Be a Mentor*, teaching is one of the best ways to learn.

The Pragmatic Bookshelf

The Pragmatic Starter Kit series: Three great titles, one objective. To get you up to speed with the essentials for successful project development. Keep your source under control, your bugs in check, and your process repeatable with these three concise, readable books.

Facets of Ruby series: Learn all about developing applications using the Ruby programming language, from the famous Pickaxe book to the latest books featuring Ruby On Rails.

The Pragmatic Bookshelf features books written by developers for developers. The titles continue the well-known Pragmatic Programmer style, and continue to garner awards and rave reviews. As development gets more and more difficult, the Pragmatic Programmers will be there with more titles and products to help programmers stay on top of their game.

Visit Us Online

My Job Went to India
`pragmaticprogrammer.com/titles/mjwti`
This book's home page, including errata and other resources.

Register for Updates
`pragmaticprogrammer.com/updates`
Be notified when updates and new books become available.

Join the Community
`pragmaticprogrammer.com/community`
Read our weblogs, join our online discussions, participate in our mailing list, interact with our wiki, and benefit from the experience of other Pragmatic Programmers.

New and Noteworthy
`pragmaticprogrammer.com/news`
Check out the latest pragmatic developments in the news.

Buy the Book

If you liked this PDF, perhaps you'd like to have a paper copy of the book. It's available for purchase at our store: pragmaticprogrammer.com/titles/mjwti.

Contact Us

Phone Orders:	1-800-699-PROG (+1 919 847 3884)
Online Orders:	www.pragmaticprogrammer.com/catalog
Customer Service:	`orders@pragmaticprogrammer.com`
Non-English Versions:	`translations@pragmaticprogrammer.com`
Pragmatic Teaching:	`academic@pragmaticprogrammer.com`
Author Proposals:	`proposals@pragmaticprogrammer.com`