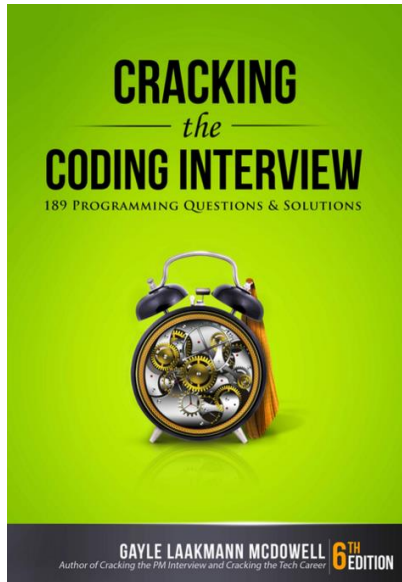


Dr. B's Programmer's Library

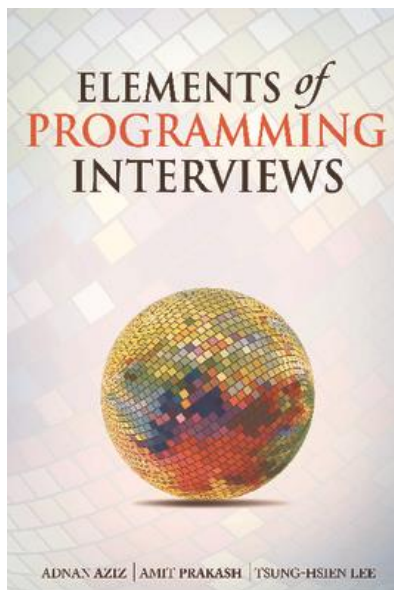
These may help you prepare for interviews and serve as a good review of material.



Cracking the Coding Interview: 150 Programming Questions and Solutions

Gayle Laakmann McDowell

<http://www.amazon.com/dp/0984782850>

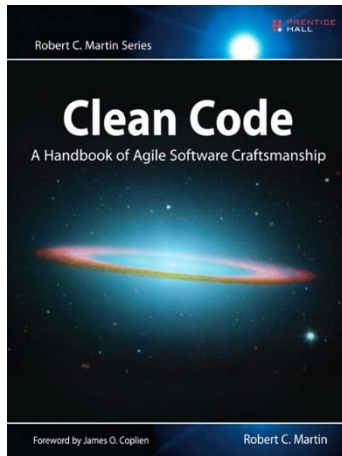


Elements of Programming Interviews: The Insiders' Guide

Adnan Aziz

<http://www.amazon.com/gp/product/1479274836>

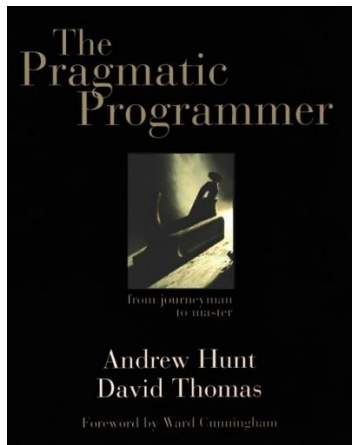
These will help you become better Programmers – a good start for your professional library.



Clean Code: A Handbook of Agile Software Craftsmanship

Robert C Martin

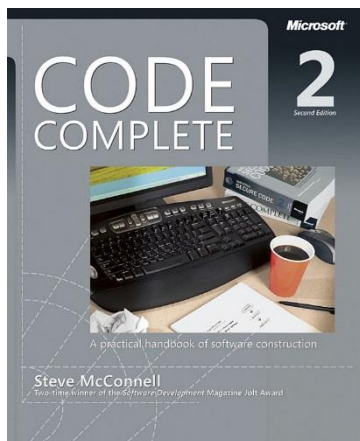
<http://www.amazon.com/Clean-Code-Handbook-Software-Craftsmanship/dp/0132350882>



The Pragmatic Programmer: From Journeyman to Master

Andrew Hunt

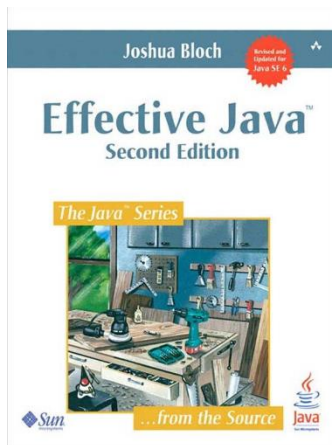
<http://www.amazon.com/dp/020161622X>



Code Complete: A Practical Handbook of Software Construction, Second Edition

Steve McConnell

<http://www.amazon.com/Code-Complete-Practical-Handbook-Construction/dp/0735619670>



Effective Java (2nd Ed)
by Joshua Bloch
(3rd Edition coming fall 2017!)

<https://www.amazon.com/Effective-Java-2nd-Joshua-Bloch/dp/0321356683>

Foreword

IF a colleague were to say to you, “*Spouse of me this night today manufactures the unusual meal in a home. You will join?*” three things would likely cross your mind: *third*, that you had been invited to dinner; *second*, that English was not your colleague’s first language; and *first*, a good deal of puzzlement. If you have ever studied a second language yourself and then tried to use it outside the classroom, you know that there are three things you must master: how the language is structured (grammar), how to name things you want to talk about (vocabulary), and the customary and effective ways to say everyday things (usage). Too often only the first two are covered in the classroom, and you find native speakers constantly suppressing their laughter as you try to make yourself understood.

It is much the same with a programming language. You need to understand the core language: is it algorithmic, functional, object-oriented? You need to know the vocabulary: what data structures, operations, and facilities are provided by the standard libraries? And you need to be familiar with the customary and effective ways to structure your code. Books about programming languages often cover only the first two, or discuss usage only spottily. Maybe that’s because the first two are in some ways easier to write about. Grammar and vocabulary are properties of the language alone, but usage is characteristic of a community that uses it.

4 Classes and Interfaces

- [Item 13: Minimize the accessibility of classes and members](#)
- [Item 14: In public classes, use accessor methods, not public fields](#)
- [Item 15: Minimize mutability](#)
- [Item 16: Favor composition over inheritance](#)
- [Item 17: Design and document for inheritance or else prohibit it](#)
- [Item 18: Prefer interfaces to abstract classes](#)
- [Item 19: Use interfaces only to define types](#)
- [Item 20: Prefer class hierarchies to tagged classes](#)
- [Item 21: Use function objects to represent strategies](#)
- [Item 22: Favor static member classes over nonstatic](#)

5 Generics

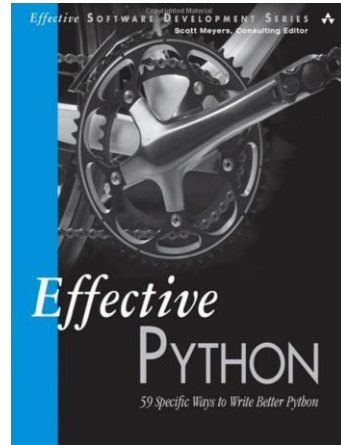
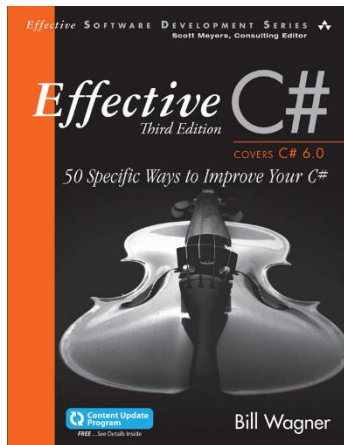
- [Item 23: Don't use raw types in new code](#)
- [Item 24: Eliminate unchecked warnings](#)
- [Item 25: Prefer lists to arrays](#)
- [Item 26: Favor generic types](#)
- [Item 27: Favor generic methods](#)
- [Item 28: Use bounded wildcards to increase API flexibility](#)
- [Item 29: Consider typesafe heterogeneous containers](#)

6 Enums and Annotations

- [Item 30: Use enums instead of `int` constants](#)
- [Item 31: Use instance fields instead of ordinals](#)
- [Item 32: Use `EnumSet` instead of bit fields](#)
- [Item 33: Use `EnumMap` instead of ordinal indexing](#)
- [Item 34: Emulate extensible enums with interfaces](#)
- [Item 35: Prefer annotations to naming patterns](#)
- [Item 36: Consistently use the `Override` annotation](#)
- [Item 37: Use marker interfaces to define types](#)

7 Methods

- [Item 38: Check parameters for validity](#)
- [Item 39: Make defensive copies when needed](#)
- [Item 40: Design method signatures carefully](#)



Effective C# (Covers C# 6.0) 50 Specific Ways to Improve Your C# (3rd Edition)

by Bill Wagner <https://www.amazon.com/Effective-Covers-Content-Update-Program/dp/0672337878>

Effective Python: 59 Specific Ways to Write Better Python by Brett

Slatkin <https://www.amazon.com/dp/0134034287>