How to develop *real life* oop skills?

Asked 16 years, 1 month ago Modified 9 years, 6 months ago

Viewed 4k times

13

I've been studying OOP for quite a while now and I have a good grasp of the theory. I read the Head First book on OOP and, while it reinforced a lot of the theory, I found the case studies to be somewhat trivial.

I find that I'm applying OOP principles to my code each day, but I'm not sure if I'm applying them correctly. I need to get to the point where I am able to look at my code and know whether I'm using inheritance appropriately, whether my object is cohesive enough, etc.

Does anyone have any good recommendations (books, online guides, blogs, walk-throughs, etc.) for taking the next step in developing solid OOP skills?

I am working primarily in .NET (visual basic), but I welcome suggestions that incorporate various platforms.

oop ooad

Share

Improve this question

Follow

edited Nov 10, 2008 at 0:53

asked Nov 9, 2008 at 22:56

Rob Sobers
21.1k ● 25 ● 85 ● 112

What language/platform? It matters because you should be looking at examples, and if we know your language we can direct you to examples. – S.Lott Nov 9, 2008 at 22:58

12 Answers

Sorted by: Highest score (default)

15

Read Refactoring by Martin Fowler, and apply it to your own work.

It will take you through a litany of malodorous characteristics of software code that describe how to detect improperly constructed classes, and even more importantly, how to fix them.

Share Improve this answer Follow

edited Nov 9, 2008 at 23:15

answered Nov 9, 2008 at 23:10

dkretz
37.6k ● 13 ● 83 ● 140

10

Consider looking into Design Patterns. Although it seems like they aren't commonly used in enterprise applications (I've seen them more commonly used in API's and Frameworks than embedded into enterprise code), they could be applied to make software simpler or more robust

in a lot of situations if only developers knew how to apply them.

The key is to understand the design patterns first, then with experience you'll learn how to apply them.

There is a Head First book on design patterns that teaches the concept pretty simply, although if you want a book that really covers design patterns in detail, check out the Gang of Four design patterns book, which is basically what made design patterns mainstream and is referred to almost every time the topic is brought up.

Design patterns can be applied in pretty much any object-oriented language to some degree or another, although some patterns can be overkill or over engineering in some cases.

EDIT:
I also want to add, you should check out the book Code Complete 2. It's a very influential book in the world of software development. It covers a lot of different concepts and theories. I learn something new every time I read it. It's such a good book that if I read it every 6 months to a year, I look at it from a different perspective that makes me a better programmer just by re-reading it. No matter how much you might think you know, this book will make you realize just how little you really know. It's really a great book. I can't stress how much you should own this book.

Follow

edited Jun 17, 2009 at 21:39

answered Nov 9, 2008 at 23:03

Dan Herbert

103k ● 51 ● 192 ● 221

5

If you already have the basics, I believe only experience will get you further. You say you are not sure if you are applying the principles correctly, but there is no one correct way. Code you write today, you'll look at in 6 months time, and wonder why you wrote it that way, and probably know of a better, cleaner way of doing it. I also guarantee that after 10 years, you'll still be learning new techniques and tricks. Don't worry too much about it, it will come, just read as much as you can, and try and apply what you read in small chunks.

Share Improve this answer

Follow

answered Nov 9, 2008 at 23:04

Steve

6,430 ● 5 ● 43 ● 66

5

I am currently half-way through the following book:

http://www.amazon.com/Applying-UML-Patterns-Introduction-Object-Oriented/dp/0131489062

I cannot recommend this book strongly enough in terms of learning a real-life, professional-grade, practical

approach to drafting and applying a well-formed and iterative design strategy before diving into code.

I, too, read the "**Head First**" book and felt that I was much better off for having read it.

After having a few years of working-world experience, I now view the Craig Larman book that I am recommending to be a perfect "next step" for me.

About the Presence of "UML" in this Book Title:

Whether you have positive feelings or negative feelings about UML notation, please do not let that influence your decision to buy the book (ISBN 0131489062) in either direction.

The prominence of "UML" in the title is misleading. While the author does use and explain UML notation, these explanations are extremely well-woven into relevant design discussions, and at no time does this book read like a boring UML spec.

In fact, here is a quote taken directly from the book:

> What's important is knowing how to think and design in objects, which is a very different and much more valuable skill than knowing UML notation. While drawing a diagram, we need to answer key questions: What are the

> responsibilities of the object? Who does it collaborate with? What design patterns should be applied? Far more important than knowing the difference between UML 1.4 and 2.0 !

This book at times seems like it is "speaking to" a lead architect or a project manager. What I mean to say by that is that it assumes that the reader has significant control over the planning and direction of a software project.

Nonetheless, even if you are only responsible for some very small piece of your company's projects and products, I would still recommend this book and encourage you to apply some "scaled down" modifications of the book's advice to your piece of the project.

Share Improve this answer Follow

edited Jun 3, 2015 at 18:11

answered Nov 9, 2008 at 23:12

pestophagous

4,233 ● 3 ● 35 ● 48

1 I also *love* this book. And I agree that the "UML" in the title is misleading. The book is about iterative development and object-oriented analysis and design. It just happens to use UML for modeling—which I also think is important, anyway; far too many developers don't know how to model abstract concepts during a discussion. – Rafa Apr 4, 2012 at 14:29

4

My OOP epiphany came from Grady Booch's book, way long time ago. Suddenly I realized *why* objects were good.

While polymorphism is cool, encapsulation is 75% of why objects are cool. It is sort of like an interface: you see the buttons but not the wiring. Before objects, only the most disciplined coders kept their grubby fingers off the internal bits of other people's procedures (it was called "structured programming").

Object make it easy to Do the Right Thing. Inheritance and polymorphism are little bonuses.

One way to learn about objects is to read other peoples' code. I learned a lot by reading the source code for the Delphi VCL framework. Even just looking at the documentation for Java will help you see what a single object class should do and how it is designed to be used by other objects.

Start a project of your own and pay attention when you want to sub-class your own classes and find that you have to go back and break up some protected methods so you can override just one piece of a process instead of replacing all of it. See how ancestors talk to descendants by calling abstract functions. In other words, go make a lot of mistakes and learn from them.

Enjoy!

Share Improve this answer

Follow

answered Nov 9, 2008 at 23:10

Richard Haven

1,155 ● 16 ● 31

3

Frankly, re-reading old David Parnas papers on information hiding helps me get in the right state of mind. The case studies may not be directly applicable but you should be able to get some useful generalizations out of them.

Share Improve this answer

Follow

answered Nov 11, 2008 at 22:23

Life is Elsewhere

51 ● 4

3

My epiphany happened when I tried to implement a very OO problem (dynamically and recursively building SQL statements) in VB6. The best way to understand polymorphism or inheritance is to need it and not be able to use it.

Share Improve this answer

Follow

answered Nov 15, 2008 at 5:51

JPLemme

4,484 ● 6 ● 32 ● 35

1

One thing that will definitely help you is working on a well-known, respected open source project. Either dig through the source code and see how things are done or try to make some additions / modifications. You'll find that there isn't one style or one right answer for most problems, but

by looking at several projects, you'll be able to get a wide view of how things can be done. From there, you'll begin to develop your own style and will hopefully make some contributions to open source in the process.

Share Improve this answer Follow

answered Nov 9, 2008 at 23:03

Jeff

Any suggestions? I'm fluent in .NET but open to other platforms and languages. – Rob Sobers Nov 9, 2008 at 23:30

Screwturn Wiki and BlogEngine .NET come to mind when thinking of open source .NET apps. Those are 2 good ones to checkout if you're interested in contributing to a C#.NET open source project. – Dan Herbert Nov 9, 2008 at 23:39

1

I think you have to attempt and fail at implementing OO solutions. That's how I did it anyway. What I mean by fail is that you end up writing smelly code while successfully delivering a working solution. After it's written you'll get a feel for where things didn't quite feel right. You may have some epiphanies, and/or you may go and hunt for a slicker solution from other programmers. Undoubtedly you'll implement some variation of standard design patterns by accident. In hindsight, a light will click on (oh! so that's what a visitor is for), and then understanding will accelerate.

As others have said, I think tooling through some good OO open source code is a good idea. So is working with

more experienced programmers who would be willing to critique your work. However understanding comes through doing.

Share Improve this answer

Follow

answered Nov 10, 2008 at 1:32

Daniel Auger

12.6k ● 5 ● 53 ● 73

> Good advice. I think that's what I'm going through right now -- delivering working solutions but thinking: "there's gotta be a better way to design this thing." – Rob Sobers Nov 10, 2008 at 4:37

1

You might want to try to read (and write) some Smalltalk for a while. Squeak is a free implementation that can show you the power of a fully object-oriented environment (unlike java or .net). All library code source is included. The language itself is incredibly simple. You'll find that java and c# are slowly adding the features well-known to Smalltalk since 1980.

Share Improve this answer

Follow

answered Nov 11, 2008 at 21:37

Stephan Eggermont

15.9k ● 1 ● 40 ● 65

0

Tortoise HG is extrodanarily well designed piece of OO open source software (written in Python).

If you already understand the basics, building something from scratch in a fully object oriented language will be a

good step in fully understanding OOP software architecture. If you don't know Python, Python Essential Reference will take you through the language in full in a few days to a week.

After you understand the language take a look through the software above and you'll have all sorts of epiphanies.

Share Improve this answer

Follow

answered Jan 25, 2010 at 16:15

orokusaki

57k ● 60 ● 183 ● 263

0

To understand basically anything thoroughly, you need to have a decent knowledge of at least one abstraction level above and one level below it. In the case of OO, others have mentioned design patterns as the layer above OO. This helps a lot to illustrate why OO is useful.

As far as the layer below OO, try to play around with higher-order functions/late binding for a while and get a feel for how these relatively simple constructs are used. Also, try to understand how OO is implemented under the hood (vtables, etc.) and how it can be done in pure C. Once you grok the value of using higher order functions and late binding, you'll quickly realize that OO is just a convenient syntax for passing around a set of related functions and the data they operate on.

Share Improve this answer

Follow

answered Jan 25, 2010 at 16:29

dsimcha

68.6k ● 55 ● 219 ● 340