# How to study design patterns? [closed]

Asked 10 years, 8 months ago Active 1 year, 1 month ago Viewed 143k times

352

I have read around 4-5 books on design patterns, but still I don't feel I have come closer to intermediate level in design patterns?

How should I go studying design patterns?

Is there any good book for design patterns?

I know this will come only with experience but there must be some way to master these?

257

design-patterns

edited May 8 '17 at 19:02

cristid9
**516** 10 19

asked Nov 24 '08 at 18:36

anand
**5,130** 20 78 142

**closed** as primarily opinion-based by Andrew Cheong, james.garriss, alko, Sebastian, Rubens Farias Nov 26 '13 at 0:13

Many good questions generate some degree of opinion based on expert experience, but answers to this question will tend to be almost entirely based on opinions, rather than facts, references, or specific expertise.

If this question can be reworded to fit the rules in the help center, please edit the question.

**locked** by Yvette Colomb ♦ Nov 30 '18 at 7:48

This question's answers are a collaborative effort. If you see something that can be improved, just edit the answer to improve it! *No additional answers can be added here.*

Read more about locked posts here.

6 The best way to learn design pattern is by doing a project. When you see patterns in the project you will know when to use them here is a article which teaches design pattern step by step with a project codeproject.com/Articles/1009532/… – Shivprasad Koirala Jul 21 '15 at 12:31

1 Take a look at the Antipatterns as well deviq.com/antipatterns – Developer Aug 30 '16 at 17:58

You can learn it from here, play.google.com/store/apps/… – Keyur Thumar Mar 13 '17 at 6:13

Probably late but still might help someone.. try geeksforgeeks.org/software-design-patterns to understand the basics and scenarios explained where they can be used. Helped me understand the foundation and purpose of each pattern – zeetit Nov 13 '17 at 12:17

See also, Pluralsight: pluralsight.com/courses/patterns-library – ssmith Feb 2 '18 at 18:47

## 22 Answers

197

The best way is to begin coding with them. Design patterns are a great concept that are hard to apply from just reading about them. Take some sample implementations that you find online and build up around them.

A great resource is the Data & Object Factory page. They go over the patterns, and give you both conceptual and real world examples. Their reference material is great, too.

answered Nov 24 '08 at 18:41

Joseph Ferris
**11.2k** 3 40 68

11 Exactly! I've always been amused that software falls under "Computer Science". I can see the argument for hardware, but software makes for a very inexact science! – Joseph Ferris Nov 24 '08 at 20:04

Sadly this resource is no longer available :(. – Citroenfris Nov 1 '14 at 13:13

5 @NielsW It's up, maybe it was only temporary down. – uthomas Nov 15 '14 at 11:35

205

I read three books and still did not understand patterns very well until I read Head First Design Patterns by OReilly. This book opened my eyes and really explained well.

edited May 15 '15 at 2:50

shareef
**5,501** 8 45 73

answered Nov 24 '08 at 18:45

Darren C
**2,121** 1 12 8

11 It's a little odd at first to be reading through a "serious" book that looks like that, but as I kept reading I noticed that I was actually understanding concepts for a change. Definitely worth a read. – Tim Whitcomb Nov 24 '08 at 18:48

16 I definitely think this is the best book for learning about design patterns. The GoF book should be used as a reference, after you understand them better. – Bill the Lizard Nov 24 '08 at 19:32

1 read this book. it's a rule – Sungguk Lim Mar 17 '10 at 14:50

10 I met Erich Gamma (one of the GoF) at a conference in Nantes, France in 2006 and he said this book was outselling the GoF book :) – Fuhrmanator Apr 26 '12 at 14:50

2 Yup. I confirm Head First is outstanding – TheEYL Jan 31 '15 at 14:19

87

My two cents for such and old question

Some people already mentioned, practice and refactoring. I believe the right order to learn about patterns is this:

1. Learn Test Driven Development (TDD)
2. Learn refactoring
3. Learn patterns

Most people ignore 1, many believe they can do 2, and almost everybody goes straight for 3.

For me the key to improve my software skills was learning TDD. It might be a long time of painful and slow coding, but writing your tests first certainly makes you think a lot about your code. If a class needs too much boilerplate or breaks easily you start noticing bad smells quite fast

The main benefit of TDD is that you lose your fear of refactoring your code and force you to write classes that are highly independent and cohesive. Without a good set of tests, it is just too painful to touch something that is not broken. With safety net you will really

adventure into drastic changes to your code. That is the moment when you can really start learning from practice.

Now comes the point where you must read books about patterns, and to my opinion, it is a complete waste of time trying too hard. I only understood patterns really well after noticing I did something similar, or I could apply that to existing code. Without the safety tests, or habits of refactoring, I would have waited until a new project. The problem of using patterns in a fresh project is that you do not see how they impact or change a working code. I only understood a software pattern once I refactored my code into one of them, never when I introduced one fresh in my code.

edited Jul 1 '18 at 11:35

AZ_
**27.1k** 24 135 188

answered Apr 19 '13 at 16:34

SystematicFrank
**10.8k** 4 44 86

can u recommend some books for TDD and refactoring majorly in C++ – anand Apr 20 '13 at 3:57

2 You will have lots of problems finding quality content if you narrow down to C++. Furthermore C++ is not the language where you want to learn testing because technically, it lacks reflection, that alone makes awfully difficult building good testing tools. It is still doable, but the communities, forums, discussion and number of TDD people are a minority within C++ because of that. I have worked a lot with it, but despite its strengths, it is not a test friendly language. – SystematicFrank Apr 20 '13 at 9:37

3 +1 for talking about the importance of TDD in yet another context. I should definitely start applying it in all of my active projects. – Gabriel C. Troia Feb 12 '14 at 18:07

59

Derek Banas made youtube tutorials for desing patterns that I like a lot:

http://www.youtube.com/playlist?list=PLF206E906175C7E07

They can be a little short in time, but his timing and presentation makes them very enjoyful to learn.

answered Jun 2 '13 at 14:26



Pedro Duarte
**705** 5 7

35

Practice, practice, practice.

You can read about playing the cello for years, and still not be able to put a bow to instrument and make anything that sounds like music.

Design patterns are best recognized as a high-level issue; one that is only relevant if you have the experience necessary to recognize them as useful. It's good that you recognize that they're useful, but unless you've seen situations where they would apply, or have

them as useful. It's good that you recognize that they're useful, but unless you've seen situations where they would apply, or have applied, it's almost impossible to understand their true value.

Where they become useful is when you recognize design patterns in others' code, or recognize a problem in the design phase that fits well with a pattern; and then examine the formal pattern, and examine the problem, and determine what the delta is between them, and what that says about both the pattern and the problem.

It's really the same as coding; K&R may be the "bible" for C, but reading it cover-to-cover several times just doesn't give one practical experience; there's no replacement for experience.

answered Nov 24 '08 at 18:55

Paul Sonier
**33.7k** 3 66 109

5 +1. I think a lot of novices jump too quickly into design patterns and start designing systems built around abstract factories, singletons, observers, visitors, etc. just straight from the book. The result is often heavy-handed, does not make the best use of the language, and not even that well-engineered from a basic coupling/cohesion standpoint (the latter especially suffers when design patterns are implemented poorly). It takes experience to decide where design patterns are appropriate, and even more to decide how to most appropriately implement them in a particular language. – stinky472 Mar 31 '11 at 2:47

25

Practice practice practice. I think 4 to 5 books is even an excessive reading exercise without some good amount of practising. Best way to do this, I believe, is to start **refactoring your current projects** using the patterns. Or if you don't have any projects you're actively working on then **just do it your own way and then try refactoring to patterns**.

You cannot appreciate them fully if you haven't suffered through the problems they solve. And please keep in mind that they are not silver bullets - you don't need to memorize them and push it hard to apply on the fly. My two cents..

edited Nov 24 '08 at 19:00

answered Nov 24 '08 at 18:51

utku_karatas
**4,247** 3 35 50

14

Ask yourself these questions:

What do they do?

What do they decouple/couple?

When should you use them?

When should you not use them?

What missing language feature would make them go away?

What technical debt do you incur by using it?

Is there a simpler way to get the job done?

answered Dec 8 '11 at 4:25

gtrak
**4,478** 3 26 41

14 and finally ask yourself from where to get answers of all above mentioned questions – Jatin Dhoot May 31 '15 at 15:39

@JatinDhoot by thinking.. – gtrak Oct 19 '15 at 16:07

8

I have found that it is a bit hard to comprehend or understand the benefits of some patterns until one understands the problems they solve and the other (worse) ways the problems have been implemented.

Other than the GOF and POSA books I have not really read any so I can't give you other recommendations. Really you just have to have an understanding of the problems domains and I think that many less experienced developers may not be able to appreciate the benefits of patterns. This is no slight against them. It is a lot easier to embrace, understand and appreciate good solutions when one has to struggle with poor alternatives first.

Good luck

answered Nov 24 '08 at 18:54

Tim
**15.7k** 19 105 204

+1 the fastest way to learn is to make the mistakes for yourself. – stinky472 Mar 31 '11 at 2:43

+1 for understanding the problems they're intended to solve. The challenge is to see the problems first-hand (on a real project that has personal importance). Too many example problems in books are (over) simplified. One of the reasons I like the book Head First Design Patterns is that they show some of the problems with the naive solutions, and how untenable those solutions are. Then they present the pattern and how clean it is... Check out Decorator in that book, for example. – Fuhrmanator Apr 26 '12 at 14:58

8

Lot of good examples have been given. I'd like to add one:

Misapply them. You don't need to do that intentionally, it will happen when you try to apply them in your initial Design-Pattern-fit. During that time every single problem that you'll see will seem to fit exactly one design pattern. Often the problems all seem to fit the same design pattern for some reason (Singelton is a primary candidate for that).

And you'll apply the pattern and it will be good. And some months later you will need to change something in the code and see that using that particular pattern wasn't that smart, because you coded yourself into a corner and you need to refactor again.

Granted, that's not really a do-that-and-you'll-learn-it-in-21-days answer, but in my experience it's the most likely to give you a good insight into the matter.

answered Nov 24 '08 at 18:56

Joachim Sauer
**244k** 50 493 567

6

Have you read "Design Patterns Explained", by Allan Shalloway.

This book is very different from other design pattern books because it is not so much a catalog of patterns, but primarily presents a way of decomposing a problem space that maps easily to patterns.

Problems can be decomposed into two parts: things that are common and things that vary. Once this is done, we map the common things to an interface, and the things that vary to an implementation. In essence, many patterns fall into this "pattern".

For example in the Strategy pattern, the common things are expressed as the strategy's context, and the variable parts are expressed as the concrete strategies.

I found this book highly thought provoking in contrast with other pattern books which, for me, have the same degree of excitement as reading a phone book.

answered Dec 3 '08 at 3:45

Phillip Ngan
**8,520** 4 49 63

5

Have you tried the Gang of Four book?

Design Patterns: Elements of Reusable Object-Oriented Software

answered Nov 24 '08 at 18:44

Rohit

**1,065** 5 16 28

7 I wouldn't recommend that book as an "eye-opening" book :) – Geo Nov 24 '08 at 18:51

69 I would recommend it as an eye-closing book. A few pages of this tome before bedtime and your insomnia will be a thing of the past. – Dónal May 11 '10 at 8:16

2 came here after when i felt asleep in the middle of the day while reading this book. searched for 'understanding design patterns'. found the post. found the comment above. made my day. agree to others, its an eye-closing book – aimme Aug 28 '16 at 10:47

5

For books, I would recommend Design Patterns Explained, and Head First Design patterns. To really learn these patterns, you should look at your existing code. Look for what patterns you are already using. Look at code smells and what patterns might solve them.

answered Nov 24 '08 at 19:00

David Nehme

**18.3k** 6 66 110

4

I've lead a few design patterns discussion groups (our site) and have read 5 or 6 patterns books. I recommend starting with the Head First Design Patterns book and attending or starting a discussion group. The Head First book might look a little Hasboro at first, but most people like it after reading a chapter or two.

Use the outstanding resource - Joshua Kereivisky's A Learning Guide to Design Patterns for the pattern ordering and to help your discussion group. Out of experience the one change I suggest to the ordering is to put Strategy first. Most of today's developers have experienced some good or bad incarnation of a Factory, so starting with Factory can lead to a lot of conversation and confusion about the pattern.This tends to take focus off how to study and learn patterns which is pretty essential at that first meeting.

answered Nov 24 '08 at 19:00

JB Brown

**589** 4 9

3

I recommend HeadFirst DesignPattern. Reading the book is not enough, after assimilating the concepts you need to findout the answers for lot of questions arise in your mind and try to findout the real life applications where in these patterns can be used. I am doing the same and started asking questions even those questions look silly.

answered Dec 15 '11 at 13:38

Rajkumar Vasan
**415** 5 21

---

2

My suggestion would be a combination of implement a few of them and analyze some implementations of them. For example, within .Net, there are uses of adapter patterns if you look at Data Adapters, as well as a few others if one does a little digging into the framework.

answered Nov 24 '08 at 18:48

JB King
**11.7k** 3 32 48

---

2

I don't know about best book, but the purists might say Design Patterns: Elements of Reusable Object-Oriented Software

As far as my personal favorite, I like Head First Design Patterns published by O'Reilly. It's written in a conversational voice that appeals to me. When I read it, I reviewed my source code at the same time to see if it applied to what I was reading. If it did, I refactored. This is how I learned Chain of Responsibility.

Practice - Practice - Practice.

answered Nov 24 '08 at 18:54

Jason Slocomb
**2,330** 17 23

---

2

Design patterns are just tools--kind of like library functions. If you know that they are there and their approximate function, you can go dig them out of a book when needed.

There is nothing magic about design patterns, and any good programmer figured 90% of them out for themselves before any books came out. For the most part I consider the books to be most useful at simply defining names for the various patterns so we can discuss them more easily.

answered Nov 24 '08 at 19:04

Bill K

**54.6k** 13 87 140

2

The way I learned design patterns is by writing lots of really terrible software. When I was about 12, I have no idea what was good or bad. I just wrote piles of spaghetti code. Over the next 10 years or so, I learned from my mistakes. I discovered what worked and what didn't. I independently invented most of the common design patterns, so when I first heard what design patterns were, I was very excited to learn about them, then very disappointed that it was just a collection of names for things that I already knew intuitively. (that joke about teaching yourself C++ in 10 years isn't actually a joke)

Moral of the story: write lots of code. As others have said, practice, practice, practice. I think until you understand why your current design is bad and go looking for a better way, you won't have a good idea of where to apply various design patterns. Design pattern books should be providing you a refined solution and a common terminology to discuss it with other developers, not a paste-in solution to a problem you don't understand.

answered Nov 24 '08 at 19:13

rmeador

**19.7k** 14 51 90

2

The notion that read design patterns, practice coding them is not really going to help IMO. When you read these books 1. Look for the basic problem that a particular design pattern solves,starting with Creational Patterns is your best bet. 2. I am sure you have written code in past, analyze if you faced the same problems that design patterns aim at providing a solution. 3. Try to redesign/re factor code or perhaps start off fresh.

About resources you can check these

1. www.dofactory.com
2. Design Patterns: Elements of Reusable Object-Oriented Software (Addison-Wesley Professional Computing Series) by Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides
3. Patterns of Enterprise Application Architecture by Martin Fowler

1 is quick start, 2 will be in depth study..3 will explain or should make you think what you learnt in 2 fits in enterprise software.

My 2 cents...

answered Nov 24 '08 at 19:56

Perpetualcoder
**10.1k** 8 55 96

1

I would think it is also difficult to study design patterns. You have to know more about OOP and some experiences with medium to big application development. For me, I study as a group of developers to make discussion. We follow A Learning Guide To Design Patterns that they have completed the patterns study. There are C# and JavaScript developers join together. It is fancy thing for me is the C# developer write codes in JavaScript and the JavaScript developer do the same thing for C# codes. After I leave a meeting I also research and read a few books at home to review. The better way to understand more and remember in my mind is to do blogging with examples in both C# and JavaScript in here http://tech.wowkhmer.com/category/Design-Patterns.aspx.

I would suggest first before going to each design patterns please understand the name of patterns. In addition if someone know the concept please just explain and give one example not only just programming but in the read world.

for example:

Factory Method:

Read world: I just give money $5, $10 or $20 and it will produce pizza back without knowing anything about how it produce, I just get a small, medium or big pizza depend on money input so that I can eat or do whatever.

Programming: The client just pass parameter value $5, $10 or $20 to the factory method and it will return Pizza object back. So the client can use that object without knowing how it process.

I'm not sure this can help you. It depends on knowledge level of people join in the meeting.

answered Nov 27 '08 at 4:39

Vorleak Chy
**389** 6 12

Second link in answer is dead. – Pang Apr 27 '17 at 6:58

1

I think you need to examine some of the issues you have encountered as a developer where you pulled your hair out after you had to revise your code for the 10th time because of a yet another design change. You probably have a list of projects where you felt that there was a lot of rework and pain.

From that list you can derive the scenarios that the Design Patterns intend to solve. Has there been a time where you needed to perform

From that list you can derive the scenarios that the Design Patterns intend to solve. Has there been a time where you needed to perform the same series of actions on different sets of data? Will you need to be able to future capability to an application but want to avoid reworking all your logic for existing classes? Start with those scenarios and return to the catalog of patterns and their respective problems they are supposed to solve. You are likely to see some matches between the GoF and your library of projects.

answered Dec 4 '08 at 2:18

David Robbins
**7,871** 7 45 79

---

1

For a beginner, Head First Design patterns would do, once we are familiar with all the patterns, then try to visualise the real time objects into those patterns.

Book will help you understand the basic concepts, unless until you have implemented in the real world you CANT Be a MASTER of the DESIGN PATTERNS

answered Feb 23 '10 at 8:51

gmhk
**7,978** 23 76 105

---