Explaining refactoring [closed]

Asked 16 years, 2 months ago Modified 11 years, 4 months ago

Viewed 721 times

8

Closed. This question does not meet Stack Overflow guidelines. It is not currently accepting answers.

This question does not appear to be about programming within the scope defined in the help center.

Closed 11 years ago.

Improve this question

Question

My question is how can you teach the methods and importance of tidying-up and refactoring code?

Background

I was recently working on a code review for a colleague. They had made some modifications to a long-gone colleagues work. During the new changes, my colleague had tried to refactor items but gave up as soon as they hit a crash or some other problem (rather than chasing the rabbit down the hole to find the root of the issue) and so reimplemented the problem code and built more on top of

that. This left the code in a tangle of workarounds and magic numbers, so I sat down with them to go through refactoring it.

I tried to explain how I was identifying the places we could refactor and how each refactoring can often highlight new areas. For example, there were two variables that stored the same information - why? I guessed it was a workaround for a bigger issue so I took out one variable and chased the rabbit down the hole, discovering other problems as we went. This eventually led to finding a problem where we were looping over the same things several times. This was due in no small part to the use of arrays of magic number sizes that obfuscated what was being done - fixing the initial "double-variable" problem led to this discovery (and others).

As I went on this refactoring journey with my colleague, it was evident that she wasn't always able to grasp why we made certain changes and how we could be sure the new functionality matched the original, so I took the time to explain and prove each change by comparing with earlier versions and stepping through the changes on paper. I also explained, through examples, how to tell if a refactoring choice was a bad idea, when to choose comments instead of code changes, and how to select good variable names.

I felt that the process of sitting together to do this was worthwhile for both myself (I got to learn a bit more about

how best to explain things to others) and my colleague (they got to understand more of our code and our coding practices) but, the experience led me to wonder if there was a better way to teach the refactoring process.

...and finally...

I understand that what does or does not need refactoring, and how to refactor it are very subjective so I want to steer clear of that discussion, but I am interested to learn how others would tackle the challenge of teaching this important skill, and if others here have had similar experiences and what they learned from them (either as the teacher or the student).

language-agnostic **refactoring**

Share

Improve this question

Follow

edited Aug 23, 2013 at 15:18

user1228

asked Oct 23, 2008 at 17:46

Jeff Yates

62.3k ● 20 ● 142 ● 192

1 This question appears to be off-topic because it is not within the bounds of discussion as described in the help center.
– user1228 Aug 23, 2013 at 15:20

9 Answers

Sorted by: Highest score (default)

3

Like most programming, refactoring skill comes with practice and experience. It would be nice to think it can be taught, but it has to be learned - and there is a significant difference in the amount of learning that can be accomplished in different environments.

To answer your question, you can teach refactoring methods and good design in a pedagogical fashion, and that's fine. But, ultimately, you and I both know attaining a certain level is only through long hard experience.

Share Improve this answer

Follow

answered Oct 23, 2008 at 18:06

Cade Roux

89.6k ● 40 ● 184 ● 266

So true. Experience is certainly a factor. A colleague of mine also believes there's an element of talent involved and that some are "born to refactor". – Jeff Yates Oct 23, 2008 at 18:08

2

I am not 100% to understand your question but I think you can refer yourself to Code Smell that need to be refactored.It contain a lot of example that you could show to other.

Here is a list of when refactoring should be used (list of code smell)

Share Improve this answer

Follow

answered Oct 23, 2008 at 17:50

Patrick Desjardins

141k ● 89 ● 294 ● 346

This is certainly useful, but I'm not convinced it's the best way of teaching someone the concepts and skills required when refactoring. – Jeff Yates Oct 23, 2008 at 18:11

At my University (Canada) it was the way they show us why refactoring is important and how to develop ability with by comparing to those smells. – Patrick Desjardins Oct 23, 2008 at 18:16

Yes. It's an important part of learning refactoring (hence the up vote). – Jeff Yates Oct 23, 2008 at 18:17

2

If you haven't read it, Martin Fowler has an excellent book on the subject called Refactoring: Improving the Design of Existing Code. He goes into substantial detail about how and why a specific piece of code should be refactored.

I hesitated to even mention it for fear that knowledge of this book is assumed by someone asking about

refactoring, and that you would think, "Duh, I meant *besides* the Fowler book." But what the hey, there you go. :-)

Share Improve this answer

Follow

answered Oct 23, 2008 at 17:59

Brian Sullivan

28.5k ● 23 ● 79 ● 92

> Thanks, I know about the book. Like I said, I don't want to discuss what should be refactored and why, I want to discuss how you teach someone that. The book certainly helps, but I think there are other methods. – Jeff Yates Oct 23, 2008 at 18:05

2

You don't mention tests. To 'prove' that a refactoring does not break the existing functionality you need to either have existing tests or write tests before doing the refactoring.

Share Improve this answer

Follow

answered Oct 23, 2008 at 18:00

Paul Croarkin

14.7k ● 16 ● 81 ● 119

> Yes, I understand that testing is important when refactoring, but I want to teach someone the importance of refactoring and how to do it. While tests are an element of refactoring, they aren't necessarily how you teach it. – Jeff Yates Oct 23, 2008 at 18:06

1

Pair Programming seems to be the best way for me to get this across. This way, as we're working on real, production code, and we both encounter some code that doesn't smell right, we tackle a code refactoring together. The pair acts as the driver's conscience saying to do the right thing instead of the quick fix, and in turn, they both learn what good code looks like in the process.

Refactoring can be an art, and just takes practice. The more you do it, the better you get at it. Keep studying the methods described in Martin Fowler's Ractoring book, and use your tools (Resharper for Visual Studio folk)

Share Improve this answer

Follow

answered Oct 23, 2008 at 17:53

casademora

69.5k ● 18 ● 71 ● 78

> This approach was certainly helpful last night. I don't think the project lead will authorise it as an ongoing approach to our work but I'm inclined to use it again in similar circumstances. – Jeff Yates Oct 23, 2008 at 18:11

1

One simple way to conceive of refactoring is right there in the name -- it's just like when you factor a common variable out of an equation:

```
xy + xz
```

becomes

```
x(y + z)
```

The x has been factored out. Refactoring code is the same thing, in that you're finding duplicate code or logic and factoring it out.

Share Improve this answer

Follow

answered Oct 23, 2008 at 18:13

dirtside

8,272 ● 11 ● 46 ● 55

Thanks. That's a great way of explaining what refactoring is doing, but it doesn't go so far as to teach the actions of refactoring and the practices therein. – Jeff Yates Oct 23, 2008 at 18:16

1 I would edit this answer so that it used `•`. – Brad Gilbert Oct 24, 2008 at 16:11

1

It sounds like your approach is a very good one. At the end of the process, you showed how you were able to uncover and fix a lot of problems. For educational purposes, it could then be interesting to invent a new change/enhancement/fix. You could then ask your mentoree how they would enact that change with the old a new codebase. Hopefully they'll see that it's much easier to make the new change with the refactored code (or how doing more refactoring would be the easiest way to prepare for the hypothetical change).

Share Improve this answer

answered Oct 23, 2008 at 18:16

Follow

Mr Fooz
112k • 7 • 76 • 103

1

I see a couple of different ways you could try to teach refactoring:

Given textbook-like examples. A downside here is that you may have contrived or simplistic examples where why refactoring is useful doesn't necessarily shine through as well as in other cases.

Refactoring existing code. In this case you could take existing legacy code that you'd clean up, or your own code in development and show the before and after doing various bits to see how much better the after is, in terms of readability and ease of maintanence. This may be a better exercise to do as this is live code being improved and enhanced to some extent.

It isn't something that someone can pick up instantly, it takes time, practice, effort and patience as some refactorings may be done for personal preference rather than because the code runs optimally one way or another.

Share Improve this answer

Follow

answered Oct 23, 2008 at 18:22

JB King
11.9k • 4 • 40 • 49

1

Teaching someone to refactor when they aren't a natural is a tough job. In my experience your best bet is to sit down with them in an office and refactor some code. While you are doing this keep up a "stream of consciousness" dialog. Talk about what you see, why the code doesn't smell right, options to refactor to, etc. Also you should make sure they're doing the same thing. The most important thing is to impart why, not how, to change the code. Any decent programmer can make a change and have it work, but it takes skill and experience to be able to state why the new solution is better than the previous.

Share Improve this answer

Follow

answered Oct 23, 2008 at 18:23

Bryan Anderson

16.1k ● 8 ● 72 ● 83