What's your most controversial programming opinion?

Asked 15 years, 11 months ago Modified 1 year, 11 months ago

Viewed 312k times

363

votes

Locked. This question and its answers are locked because the question is off-topic but has historical significance. It is not currently accepting new answers or interactions.

This is definitely subjective, but I'd like to try to avoid it becoming argumentative. I think it could be an interesting question if people treat it appropriately.

The idea for this question came from the comment thread from my answer to the "What are five things you hate about your favorite language?" question. I contended that classes in C# should be sealed by default - I won't put my reasoning in the question, but I might write a fuller explanation as an answer to this question. I was surprised at the heat of the discussion in the comments (25 comments currently).

So, what contentious opinions do *you* hold? I'd rather avoid the kind of thing which ends up being pretty religious with relatively little basis (e.g. brace placing) but examples might include things like "unit testing isn't actually terribly

helpful" or "public fields are okay really". The important thing (to me, anyway) is that you've got reasons behind your opinions.

Please present your opinion and reasoning - I would encourage people to vote for opinions which are well-argued and interesting, whether or not you happen to agree with them.

language-agnostic

Share

edited May 23, 2017 at 12:10

community wiki
6 revs, 4 users 61%
Jon Skeet

Comments disabled on deleted / locked posts / reviews

407 Answers

Sorted by: Highest score (default)

3

votes

Anonymous functions suck.

I'm teaching myself jQuery and, while it's an elegant and immensely useful technology, most people seem to treat it as some kind of competition in maximizing the user of anonymous functions.

Function and procedure naming (along with variable naming) is the greatest expressive ability we have in programming. Passing functions around as data is a great technique, but making them anonymous and therefore non-self-documenting is a mistake. It's a lost chance for expressing the meaning of the code.

Share

answered Oct 14, 2009 at 18:19

community wiki
Larry Lustig

2 While I haven't used jQuery, I have to disagree with the general principle. The ability to express (say) a projection or a filter *right where you're using it* rather than having to introduce a separate function is one of the nicest features in C# 2 and 3. (Nicer in 3 than 2, as lambda expressions are neater than anonymous methods.) – Jon Skeet Oct 14, 2009 at 20:23

@Jon: Well, I guess that officially makes my opinion controversial. I still disagree. While it's nice to be able to express functionality that way, for all but the most trivial cases it fundamentally detracts from the readability of the code. If you could name the function in place that would help with the issue of expressing your purpose, but it still wouldn't eliminate the problem of actually reading functions nested in the parameter list of other functions which, in turn, are often nested inside another functions parameter list. – Larry Lustig Oct 14, 2009 at 20:36

You can name inline functions in JavaScript, if you want to. Just include a name between "function" and the arguments: var s = function square(a) { return a * a; }; – Mark Bessey Oct 23, 2009 at 21:27

3 votes

Never change what is not broken.

Share

answered Oct 16, 2009 at 19:35

community wiki
Varma

1 What if it works, but is unmaintanable, ugly, difficult to understand and likely to break if something else changes? – simon Oct 19, 2009 at 9:14

That is the exact reason why I posted this as "controversial". – Varma Oct 20, 2009 at 5:06

"Refactor Mercilessly". XP Manifesto. But only if you have comprehensive unit tests in place... – Engineer Sep 17, 2010 at 15:43

3 votes

Detailed designs are a waste of time, and if an engineer needs them in order to do a decent job, then it's not worth employing them!

OK, so a couple of ideas are thrown together here:

1) the old idea of waterfall development where you supposedly did all your design up front, resulting in some glorified extremely detailed class diagrams, sequence diagrams etc. etc., was a complete waste of time. As I once said to a colleague, I'll be done with design once the code is finished. Which I think is what agile is partly a recognition of - that the code is the design, and that any decent developer is continually refactoring. This of course, makes the idea that your class diagrams are out of date laughable - they always will be.

2) management often thinks that you can usefully take a poor engineer and use them as a 'code monkey' - in other words they're not particularly talented, but heck - can't you use them to write some code. Well.. no! If you have to

spend so much time writing detailed specs that you're basically specifying the code, then it will be quicker to write it yourself. You're not saving any time. If a developer isn't smart enough to use their own imagination and judgement they're not worth employing. (Note, I'm not talking about junior engineers who are able to learn. Plenty of 'senior engineers' fall into this category.)

Share

edited Oct 18, 2009 at 3:33

community wiki
2 revs, 2 users 80%
Phil

++ I liken spec-writing to driving a car at night in a fog. You can only see so far ahead, and turning up the brightness of the lights does not help. The supply of information is simply limited. It's worth getting as much as you can, but what you really have to be able to do is adapt when more information becomes available as you proceed. – Mike Dunlavey Oct 23, 2009 at 21:04

... I was once handed a design like that. The design doc was about 2 inches thick in paper and projected to take 18 mm to develop. I talked them into writing a code-generator. The *final source* was 1/2 inch thick, was done in 4 mm, and had blazing performance. – Mike Dunlavey Oct 23, 2009 at 21:07

... That's why I believe in prototyping, rapid or not. When I'm developing some new product, I like to be able to do at least 3 throw-away versions, because that's how I can see deeper into the fog. Good post! – Mike Dunlavey Oct 30, 2009 at 14:50

Thanks Mike, and agree with what you're saying - it's impractical to expect to be able to get all the design right up front - you've got to 'try something', then rework it as you discover more about the requirements and both how best to implement it, and often also how the technologies you're using are best used. – Phil Oct 30, 2009 at 19:39

3 votes

If you have ever let anyone from rentacoder.com touch your project, both it and your business are completely devoid of worth.

Share

answered Oct 18, 2009 at 3:40

community wiki
Azeem.Butt

Not controversial, this is a statement of fact. – mynameiscoffey May 19, 2010 at 1:25

3 votes

I have two:

Design patterns are sometimes a way for bad programmer to write bad code - "when you have a hammer - all the world looks like a nail" mentality. If there si something I hate to hear is two developers create design by patterns: "We should use command with facade ...".

There is no such thing as "premature optimization". You should profile and optimize the your code before you

get to that point when it becomes too painful to do so.

Share

answered Oct 22, 2009 at 15:39

community wiki
Dror Helper

1 Premature optimization does indeed exist and is very much a problem. With very few exceptions, your goal is to satisfy a function as per business requirements. Make it work, make it right, then make it faster. Optimizing without understanding the whole application profile is like throwing money out of a window. Let me know where you work, because I'll be downstairs with a net to catch some of it. ;-) – Joseph Ferris Oct 29, 2009 at 19:09

You're right - but only some of the time... I've seen the "premature optimization" card used way too many time to create a bad very hard to improve application flow. If you can write it better the first time, why not do so? – Dror Helper Oct 29, 2009 at 20:30

1 I think the best rule is to always make things as simple as possible. It is much easier to optimize simple code than to simplify optimized code. – thesmart Nov 4, 2009 at 2:25

3 votes

There is only one design pattern: encapsulation

For example:

- Factory method: you've encapsulated object creation

- Strategy: you encapsulated different changeable algorithms
- Iterator: you encapsulated the way to sequentially access the elements in the collection.

Share

answered Nov 1, 2009 at 12:29

community wiki
flybywire

1 wrong. the only design pattern is "take out duplicate code and put it in an external function/method/object" – hasen Nov 13, 2009 at 21:31

3

votes

Java is the COBOL of our generation.

Everyone learns to code it. There code for it running in big companies that will try to keep it running for decades. Everyone comes to despise it compared to all the other choices out there but are forced to use it anyway because it pays the bills.

Share

answered Nov 9, 2009 at 19:28

community wiki
Kelly S. French

3 COBOL is still the COBOL of our generation. Maybe Java will be the COBOL three generations from now... But then, so will C#. – Kobi Nov 15, 2009 at 7:13

I would say PHP is the COBOL of our generation. It has an important property in common - it was designed to be coded by people who were not full-time coders. Unlike Java and C# which borrow heavily from C++. – finnw Feb 11, 2010 at 16:57

3 votes

Macros, **Preprocessor** instructions and **Annotations** are evil.

One syntax and language per file please!

// does not apply to Make files, or editor macros that insert real code.

Share

answered Nov 18, 2009 at 20:51

community wiki
Chris Cudmore

Everyone agrees that the pre-processor is evil... except the people who would never be found on Stack Overflow. They love it. – Integer Poet Mar 15, 2010 at 19:26

How about this: C is evil. And C++ is even more evil. However, C is a necessary evil, and C++ an unnecessary one.
– Warren P Apr 1, 2010 at 0:05

3

votes

Storing XML in a CLOB in a relational database is often a horrible cop-out. Not only is it hideous in terms of performance, it shifts responsibility for correctly managing structure of the data away from the database architect and onto the application programmer.

Share

answered Dec 9, 2009 at 20:02

community wiki
Tim

3

votes

Development is 80% about the design and 20% about coding

I believe that developers should spend 80% of time designing at the fine level of detail, what they are going to build and only 20% actually coding what they've designed. This will produce code with near zero bugs and save a lot on test-fix-retest cycle.

Getting to the metal (or IDE) early is like premature optimization, which is know to be a root of all evil. Thoughtful upfront design (I'm not necessarily talking about enormous design document, simple drawings on white board will work as well) will yield much better results than just coding and fixing.

Share

answered Dec 15, 2009 at 5:41

community wiki
Dima Malenko

3

votes

Size matters! Embellish your code so it looks bigger.

Share

edited Jan 4, 2010 at 19:04

community wiki
2 revs
fastcodejava

3

votes

Programmers should never touch Word (or PowerPoint)

Unless you are developing a word or a document processing tool, you should not touch a Word processor that emits only binary blobs, and for that matter:

Generated XML files *are* binary blobs

Programmers should write plain text documents. The documents a programmer writes need to convey intention only, not formatting. It must be producible with the programming tool-chain: editor, version-control, search utilities, build system and the like. When you are already have and know how to use that tool-chain, every other document production tool is a horrible waste of time and effort.

When there is a need to produce a document for non-programmers, a lightweight markup language should be used such as reStructuredText (if you are writing a plain text file, you are probably writing your own lightweight markup anyway), and generate HTML, PDF, S5, etc. from it.

Share

edited Mar 10, 2010 at 13:33

community wiki
5 revs, 2 users 90%
Chen Levy

3

votes

Women make better programmers than men.

The female programmers I've worked with don't get wedded to "their" code as much as men do. They're much more open to criticism and new ideas.

Share

edited Aug 8, 2010 at 23:53

community wiki
3 revs
WOPR

While my experience agrees with your explanation (based on only 2 data points however), I don't agree with your assessment that not being wedded to their code is all it takes to

be a better programmer. – Cameron MacFarland Jan 13, 2009 at 23:04

Good programmers form a strong emotional attachment to thier code because they're passionate about creating the best solutions they can. It almost represents the best they can be .. hence the ego attachment. It's both blinding to better solutions and a key driver to improve. IMHO; it's mostly good!
– John MacIntyre Jan 22, 2009 at 19:24

1 I've not seen a correlation between sex and code-base ownership, either. (Though only two data points, also.) Care to expand on your answer? – Stu Thompson Apr 28, 2009 at 20:11

2

votes

To Be A Good Programmer really requires working in multiple aspects of the field: Application development, Systems (Kernel) work, User Interface Design, Database, and so on. There are certain approaches common to all, and certain approaches that are specific to one aspect of the job. You need to learn how to program Java like a Java coder, not like a C++ coder and vice versa. User Interface design is really hard, and uses a different part of your brain than coding, but implementing that UI in code is yet another skill as well. It is not just that there is no "one" approach to coding, but there is not just one type of coding.

Share

answered Jan 2, 2009 at 19:30

community wiki
admiyo

2

votes

That software can be bug free if you have the right tools and take the time to write it properly.

Share

answered Jan 6, 2009 at 21:02

community wiki
too much php

> And what is about Gödel's completeness theorem?
> – Vlad Gudim Jan 7, 2009 at 14:01

Sorry I'm not familiar with that one, and I'll have to read the Wikipedia page about 10 times more to make sense of it I feel. – too much php Jan 7, 2009 at 21:27

2 votes

Opinion: **Not having function definitions, and return types can lead to flexible and readable code.**

This opinion probably applies more to interpreted languages than compiled. Requiring a return type, and a function argument list, are great for things like intellisense to auto document your code, but they are also restrictions.

Now don't get me wrong, I am not saying throw away return types, or argument lists. They have their place. And 90% of the time they are more of a benefit than a hindrance.

There are times and places when this is useful.

Share

answered Jan 7, 2009 at 6:51

community wiki
J.J.

2 votes

You can't write a web application without a remote debugger

Web applications typically tie together interactions between multiple languages on the client and server side, require interaction from a user and often include third-party code

that can be anything from a simple API implementation to a byzantine framework.

I've lost count of the number of times I've had another developer sat with me while I step into and follow through what's actually going on in a complex web application with a decent remote debugger to see them flabbergasted and amazed that such tools exist. Often they still don't take the trouble to install and setup these kinds of tools even after seeing them in action.

You just can't debug a non trivial web application with print statements. Times ten if you didn't right all the code in your application.

If your debugger can step through all the various languages in use and show you the http transactions taking place then so much the better.

You can't develop web applications without Firebug

Along similar lines, once you have used Firebug (or very near equivalent) you look on anyone trying to develop web applications with a mixture of pity and horror. Particularly with Firebug showing computed styles, if you remember back to NOT using it and spending hours randomly changing various bits of CSS and adding "!important" in too many places to be funny you will *never go back*.

answered Jan 9, 2009 at 14:19

community wiki
reefnet_alex

I agree to firebug, but ive been a web dev. for 3 years and done everything from mid to large. I've never felt the need to use a remote debugger – Ali Jan 9, 2009 at 18:03

exackly - and before you used firebug I bet you didn't realise you needed it either :) seriously though, give it a try and then say that – reefnet_alex Jan 9, 2009 at 21:06

Since I can unit test my whole webapp, why would I need a remote debugger? I can run any line of code I want locally... – Aaron Digulla Mar 3, 2009 at 14:59

1) remote doesn't mean "not local" in this case, it means it running the debugger on the php interpreter as run up by your web server and following all the interactions with the browser through. whether running locally or on a live server you need a remote debugger to see what's actually happening – reefnet_alex Mar 12, 2009 at 13:02

2) live server != dev machine: there are some bugs you will only see on your live (or exact copy of your live) server – reefnet_alex Mar 12, 2009 at 13:04

2
votes

Believe it or not, my belief that, **in an OO language, most of the (business logic) code that operates on a class's data should be in the class itself** is heresy on my team.

Share

answered Jan 9, 2009 at 22:37

community wiki

moffdub

I second your opinion. I can't stand it when someone goes with the excuse that "Classes should be minimal, clean, simple" and writes a close to useless class that merely aggregates data - and then builds the logic about this data everywhere else. – Daniel Daranas Jan 10, 2009 at 0:28

Hmm. So, the `cut` method should be a member of which class? `meat`, `vegetable`, `knife`, `scissors`, `kitchen_table`, `workbench` ...? – Svante Jan 12, 2009 at 16:47

2 Without any further information, I'd say that the knife cuts a cuttable object: Knife.cut(ICuttable something). Of course, if you only have one cuttable object, like meat, and many things that cut the meat, then you want Meat.cutWith(ICutter something). – moffdub Jan 17, 2009 at 0:54

2 votes

2 space indent.

No discussion. It just has to be that way ;-)

Share

answered Jan 12, 2009 at 9:04

community wiki
Fergie

How about a compromise? You add a space and I give up a space and everybodies happy? ;-) – Captain Sensible Jan 26, 2009 at 15:25

There was an argument among the Delphi programmers at my company about whether to use 2-space indents or 4-space indents. We settled on 3-spaces to offend all parties equally. – Juliet Feb 4, 2009 at 0:39

3-space is the ugliest indent I've ever seen. It just does not fit into my happy 2^n world. 2 is just for those who want to write code with 20-way and more indentation, i.e. for those whose application consists of one monster class. Or a monster function. – Sebastian Mach Mar 19, 2009 at 15:20

2

votes

[Code as Design: Three Essays by Jack W. Reeves]

The source code of any software is its most accurate design document. Everything else (specs, docs, and sometimes comments) is either incorrect, outdated or misleading.

Guaranteed to get you fired pretty much everywhere.

Share

answered Jan 14, 2009 at 14:30

community wiki
fbonnet

2

votes

Tcl/Tk is the best GUI language/toolkit combo ever

It may lack specific widgets and be less good-looking than the new kids on the block, but its model is elegant and so easy to use that one can build working GUIs faster **by**

typing commands interactively than by using a visual interface builder. Its expressive power is unbeatable: other solutions (Gtk, Java, .NET, MFC...) typically require ten to one hundred LOC to get the same result as a Tcl/Tk one-liner. All without even sacrificing readability or stability.

```
pack [label .l -text "Hello world!"] [button .b -
text "Quit" -command exit]
```

Share

answered Jan 16, 2009 at 8:54

community wiki
fbonnet

Oh my god is that controversial! :P – Aaron Powell Jan 18, 2009 at 10:55

I'd say WPF gives Tcl/Tk a run for it's money.
– Cameron MacFarland Jan 22, 2009 at 22:41

I suggest to have a look at pygtk and or Groovy Swingbuilder. Both concepts are more powerful than the strange Tcl syntax and achieve the same result with a modern widget set and little code. And if you want to see something really cool, goole for "enthought traits". – Aaron Digulla Mar 2, 2009 at 14:20

WPF should have been called WTF. Because hand-tuning XML is my idea of a RAD good time. – Warren P Apr 1, 2010 at 0:13

2 votes

What strikes me as amusing about this question is that I've just read the first page of answers, and so far, I haven't

found a single controversial opinion.

Perhaps that says more about the way stackoverflow generates consensus than anything else. Maybe I should have started at the bottom. :-)

Share

answered Jan 18, 2009 at 9:38

community wiki
Dominic Cronin

There's definitely a mixture of controversial and not. Look for ones with lots of comments :) – Jon Skeet Jan 18, 2009 at 10:03

haha you know what, i'll mark you down so ppl will read this when looking through these. I agree with you 100%
– user34537 Jan 26, 2009 at 7:21

lots of controversy on page 2, though :) – Juliet Feb 4, 2009 at 0:58

2 votes

There are some (very few) legitimate uses for goto (particularly in C, as a stand-in for exception handling).

Share

answered Jan 21, 2009 at 23:28

community wiki
DWright

2

votes

Dependency Management Software Does More Harm Than Good

I've worked on Java projects that included upwards of a hundred different libraries. In most cases, each library has its own dependencies, and those dependent libraries have their own dependencies too.

Software like Maven or Ivy supposedly "manage" this problem by automatically fetching the correct version of each library and then recursively fetching all of its dependencies.

Problem solved, right?

Wrong.

Downloading libraries is the **easy** part of dependency management. The hard part is creating a mental model of the software, and how it interacts with all those libraries.

My unpopular opinion is this:

If you can't verbally explain, off the top of your head, the basic interactions between all the libraries in your project, you should eliminate dependencies until you can.

Along the same lines, if it takes you longer than ten seconds to list all of the libraries (and their methods) invoked either directly or indirectly from one of your functions, then you are doing a poor job of managing dependencies.

You should be able to easily answer the question "which parts of my application actually depend on library XYZ?"

The current crop of dependency management tools do more harm than good, because they make it easy to create impossibly-complicated dependency graphs, and they provide virtually no functionality for reducing dependencies or identifying problems.

I've seen developers include 10 or 20 MB worth of libraries, introducing thousands of dependent classes into the project, just to eliminate a few dozen lines of simple custom code.

Using libraries and frameworks can be good. But there's always a **cost**, and tools which obscure that cost are inherently problematic.

Moreover, it's sometimes (note: certainly not always) better to reinvent the wheel by writing a few small classes that implement **exactly** what you need than to introduce a dependency on a large general-purpose library.

Share

edited Jan 22, 2009 at 22:11

community wiki
3 revs
benjismith

I disagree. I think DM is a good thing, except Maven did it wrong. So much code depends on other stuff, but we still after

so many year haven't figured out how to manage it all. That's one thing we need to fix for SW dev to move forward.
– Pyrolistical Mar 23, 2009 at 22:02

2 votes

Never implement anything as a singleton.

You can decide not to construct more than one instance, but always ensure you implementation can handle more.

I have yet to find any scenario where using a singleton is actually the right thing to do.

I got into some very heated discussions over this in the last few years, but in the end I was always right.

Share

answered Feb 3, 2009 at 19:40

community wiki
Andreas

I can give you one: to abstract away, transparently, support for different JVM versions, using reflection to load a support instance for J5, gracefully defaulting to one for J2 if the JVM is < J5 - e.g. getting time is nano seconds. – L. Cornelius Dol Feb 6, 2009 at 9:54

2 votes

Useful and clean high-level abstractions are significantly more important than performance

one example:

Too often I watch peers spending hours writing over complicated Sprocs, or massive LINQ queries which return unintuitive anonymous types for the sake of "performance".

They could achieve almost the same performance but with considerably cleaner, intuitive code.

Share

answered Feb 4, 2009 at 22:43

community wiki
andy

2
votes

Automatic Updates Lead to Poorer Quality Software that is Less Secure

The Idea

A system to keep users' software up to date with the latest bug fixes and security patches.

The Reality

Products have to be shipped by fixed deadlines, often at the expense of QA. Software is then released with many bugs and security holes in order to meet the deadline in the knowledge that the 'Automatic Update' can be used to fix all the problems later.

Now, the piece of software that really made me think of this is VS2K5. At first, it was great, but as the updates were

installed the software is slowly getting worse. The biggest offence was the loss of macros - I had spent a long time creating a set of useful VBA macros to automate some of the code I write - but apparently there was a security hole and instead of fixing it the macro system was disabled. Bang goes a really useful feature: recording keystrokes and repeated replaying of them.

Now, if I were really paranoid, I could see Automatic Updates as a way to get people to upgrade their software by slowly installing code that breaks the system more often. As the system becomes more unreliable, users are tempted to pay out for the next version with the promise of better reliablity and so on.

Skizz

Share

answered Mar 9, 2009 at 11:55

community wiki
Skizz

No, software is released early, before being fully tested, because it can be updated later - there is less emphasis on creating bug-free code and more on getting it released. This gives a 'window of opportunity' to malicious code. – Skizz Mar 11, 2009 at 13:46

It's not as if MS had a shining reputation for secure, bug-free software before automatic updates came along. – Nate C-K Jan 7, 2010 at 19:02

I'd actually say the reverse is true, overall security and stability have improved in more recent MS software. – Nate C-K Jan 7, 2010 at 22:08

2 votes

MS Access* is a Real Development Tool and it can be used without shame by professional programmers

Just because a particular platform is a magnet for hacks and secretaries who think they are programmers shouldn't besmirch the platform itself. Every platform has its benefits and drawbacks.

Programmers who bemoan certain platforms or tools or belittle them as "toys" are more likely to be far less knowledgable about their craft than their ego has convinced them they are. It is a definite sign of overconfidence for me to hear a programmer bash any environment that they have not personally used extensively enough to know well.

* Insert just about any maligned tool (VB, PHP, etc.) here.

Share

answered Mar 20, 2009 at 19:31

community wiki
JohnFx

I agree by proxy... a former colleague manipulated and massaged an Access-backed production system into a highly efficient system that was perfectly suited for its needs. Although with the availability of other desktop-based DB platforms such

as SQL Compact (aka SQL Compact Edition, aka SQL Mobile) Access is becomming more of a developer's occasional assistant than his tool. It's like a toothpick kind of--developers can use it, and maybe even use it professionally (give me back my CD!)... – STW Apr 21, 2009 at 20:14

I do have to disagree about PHP, at least in the pre/early asp.net days. PHP was a very valid competitor to classic ASP, and it wasn't until ASP.NET came along and IIS 6 was released that PHP began to lose its functionality. LAMP blew away IIS/asp in my opinion, and judging by the dominance of Apache servers running the web I would say the internet would more or less agree. – STW Apr 21, 2009 at 20:16

I got one agree and one disagree comment. I should at least get an upvote for that since the OP wanted controversial opinions. =) – JohnFx Apr 21, 2009 at 20:55

disagree.i decided to not to use it when i was 11.because autonumber s..ks. – Behrooz Dec 14, 2009 at 19:50

Ok, I agree, but they really should standardize the SQL. Its crap having to work with access specific queries. – JL. Apr 4, 2010 at 16:14

2

votes

...That the "clarification of ideas" should not be the sole responsibility of the developer...and yes xkcd made me use that specific phrase...

To often we are handed project's that are specified in psuedo-meta-sorta-kinda-specific "code" if you want to call it that. There are often product managers who draw up the initial requiements for a project and perform next to 0% of basic logic validation.

I'm not saying that the technical approach shouldn't be drawn up by the architect, or that the speicifc implemntation shouldn't be the responsibility of the developer, but rather that it should the requirement of the product manager to ensure that their requirements are logically feasible.

Personally I've been involved in too many "simple" projects that encounter a little scope creep here and there and then come across a "small" change or feature addition which contradicts previous requirements--whether implicitly or explicitly. In these cases it is all too easy for the person requesting the borderline-impossible change to become enraged that developers can't make their dream a reality.

Share

answered Apr 21, 2009 at 20:08

community wiki
STW

2

votes

switch-case is not object oriented programming

I often see a lot of switch-case or awful big if-else constructs. This is merely a sign for not putting state where it belongs and don't use the real and efficient switch-case construct that is already there: method lookup/vtable

Share

answered May 6, 2009 at 6:50

community wiki
Norbert Hartl