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)

Prev 1 … 5 6 7 8 9 … 14 Next

7

votes

My controversial opinion: **OO Programming is vastly overrated [and treated like a silver bullet], when it is really just *another* tool in the toolbox, nothing more!**

Share

answered Mar 27, 2009 at 16:59

community wiki
torial

7

votes

Most of programming job interview questions are pointless. Especially those figured out by programmers.

It is a common case, at least according to my & my friends experience, where a puffed up programmer, asks you some tricky wtf he spent weeks googling for. The funny thing about that is, you get home and google it within a minute. It's like they often try to beat you up with their *sophisticated weapons*, instead of checking if you'd be a comprehensive, pragmatic team player to work with.

Similar stupidity IMO is when you're being asked for highly accessible fundamentals, like: "Oh wait, let me see if you can pseudo-code that `insert_name_here`-algorithm on a sheet of paper (sic!)". Do I really need to remember it while applying for a high-level programming job? Should I efficiently solve problems or puzzles?

Share

answered Mar 30, 2009 at 12:57

community wiki
ohnoes

+1 fully agree, its also usually the case that during the interview they check to see if you are the rocket scientist they require.

Asking you all sorts of rough questions. Then we you get the job, you realize actually what they were after was a coding monkey, who shouldn't get too involved in business decisions. I know this is not always the case, but usually the work you end up doing is very easy compared to the interview process, where you would think they were looking for someone to develop organic rocket fuel. – JL. Apr 4, 2010 at 16:13

7

votes

Tools, Methodology, Patterns, Frameworks, etc. are no substitute for a properly trained programmer

I'm sick and tired of dealing with people (mostly managers) who think that the latest tool, methodology, pattern or framework is a silver bullet that will eliminate the need for hiring experienced developers to write their software. Although, as a consultant who makes a living rescuing at-risk projects, I shouldn't complain.

Share

answered Apr 29, 2009 at 18:13

community wiki
Jeff Leonard

I will second "Thou Shalt Not Complain". Those who manage based on idealistic expedience and feel good tools always find themselves in trouble like this. Unfortunately I have noticed that no matter how many times you deliver the reality that you need to use good people. The bottom line bean counters always try to find the cheap/easy way out. In the end they always have to poney up the money. They either pony up to get it done correctly the first time or they pony up to have it fied properly by someone who chanrges a premium. Sometimes far in excess

of the cost to do it right the 1st time. – Axxmasterr Jul 27, 2009 at 17:05

7

votes

The simplest approach is the best approach

Programmers like to solve assumed or inferred requirements that add levels of complexity to a solution.

"I assume this block of code is going to be a performance bottleneck, therefore I will add all this extra code to mitigate this problem."

"I assume the user is going to want to do X, therefore I will add this really cool additional feature."

"If I make my code solve for this unneeded scenario it will be a good opportunity to use this new technology I've been interested in trying out."

In reality, the simplest solution that meets the requirements is best. This also gives you the most flexibility in taking your solution in a new direction if and when new requirements or problems come up.

Share

answered Jun 11, 2009 at 16:33

community wiki
Brad C

Yeah, the best way to compare implemntations is by their line count. People wont reuse your code unless it's less than one page long. – AareP Jun 13, 2009 at 16:03

1 ++ I don't think this is controversial in one sense - everybody agrees with it. But in another sense it is controversial - because few people follow it. – Mike Dunlavey Oct 13, 2009 at 22:52

7 votes

Exceptions should only be used in truly exceptional cases

It seems like the use of exceptions has run rampant on the projects I've worked on recently.

Here's an example:

We have filters that intercept web requests. The filter calls a screener, and the screener's job is to check to see if the request has certain input parameters and validate the parameters. You set the fields to check for, and the abstract class makes sure the parameters are not blank, then calls a screen() method implemented by your particular class to do more extended validation:

```
public boolean processScreener(HttpServletRequest
req, HttpServletResponse resp, FilterConfig
filterConfig) throws Exception{
            //
            if (!checkFieldExistence(req)){
                    return false;
            }
            return screen(req,resp,filterConfig);
    }
```

That checkFieldExistance(req) method **never** returns false. It returns true if none of the fields are missing, and throws an exception if a field is missing.

I know that this is bad design, but part of the problem is that some architects here believe that you need to throw an exception every time you hit something unexpected.

Also, I am aware that the signature of checkFieldExistance(req) **does** throw an Exception, its just that almost all of our methods do - so it didn't occur to me that the method might throw an exception instead of returning false. Only until I dug through the code I noticed it.

Share

answered Jul 21, 2009 at 15:10

community wiki
LGriffel

> And don't forget the overhead involved when throwing an exception as well. Throw/catch might be fairly harmless performance-wise for a single operation, but start looping over it and... ho-boy. I speak from experience. – Tullo_x86 Sep 18, 2009 at 21:53

7

votes

Controversial eh? I reckon the fact that C++ streams use << and >>. I hate it. They are shift operators. Overloading them in this way is plain bad practice. It makes me want to

kill whoever came up with that and thought it was a good idea. GRRR.

Share

answered Oct 14, 2009 at 16:55

community wiki
Goz

7

votes

"Comments are Lies"

Comments don't run and are easily neglected. It's better to express the intention with clear, refactored code illustrated by unit tests. (Unit tests written TDD of course...)

We don't write comments because they're verbose and obscure what's really going on in the code. If you feel the need to comment - find out what's not clear in the code and refactor/write clearer tests until there's no need for the comment...

... something I learned from Extreme Programming (assumes of course that you have established team norms for cleaning the code...)

Share

answered Oct 25, 2009 at 22:11

community wiki
Dafydd Rees

3 Code will only explain the "how" something is done and not the "why". It is really important to distinguish between the two. Decisions sometimes have to be made and the reason for that decision needs to live on. I find that it is important to find a middle ground. The "no comments" crowd are just as much cultists as "comment everything" crowd. – Joseph Ferris Oct 29, 2009 at 19:07

You're right about this: "Code will only explain the "how" something is done" If I want to know what it does, I'll find the TDD-written test that's covering it. If there's a mystery about what it does and it's important enough, I'll insert a breakage (e.g. throw new RuntimeException("here it is")) and run all the acceptance tests to see what scenarios need that code path to run. – Dafydd Rees Oct 29, 2009 at 19:28

Thia is why i said comments are evil in my post stackoverflow.com/questions/406760/… I am proud my answer is the most serious most downvoted answer :) – user34537 Nov 20, 2009 at 12:09

If you want to know why something is running, just inject a bug e.g. throw new RuntimeException("HERE"); into it and run the functional tests. Read off the names of the failing system-level tests - that's why you need that piece of code. – Dafydd Rees Nov 20, 2009 at 15:24

No, that's just more what. Good comments explain why the function works THE WAY it does, not why it exists, which is ultimately just a what. – Integer Poet Mar 15, 2010 at 19:42

---

7 votes

Modern C++ is a beautiful language.

There, I said it. A lot of people really hate C++, but honestly, I find modern C++ with STL/Boost style programming to be

a very expressive, elegant, and incredibly productive language most of the time.

I think most people who hate C++ are basing that on bad experiences with OO. C++ doesn't do OO very well because polymorphism often depends on heap-allocated objects, and C++ doesn't have automatic garbage collection.

But C++ really shines when it comes to generic libraries and functional-programming techniques which make it possible to build incredibly large, highly-maintainable systems. A lot of people say C++ tries to do everything, but ends up doing nothing very well. I'd probably agree that it doesn't do OO as well as other languages, but it does generic programming and functional programming ***better*** than any other mainstream C-based language. (C++0x will only further underscore this truth.)

I also appreciate how C++ lets me get low-level if necessary, and provides full access to the operating system.

Plus RAII. Seriously. I really miss destructors when I program in other C-based languages. (And no, garbage collection does not make destructors useless.)

edited Nov 1, 2009 at 13:34

community wiki

3 revs
Charles Salvia

1 I really dislike the C++ compilers. They have terrible error messages. – thesmart Nov 4, 2009 at 2:21

"any mainstream C-based language" would include C# and Scala, both of which are now quite good for functional programming. You should look at them again if you haven't tried the latest versions yet. – finnw Feb 11, 2010 at 17:04

7 votes

JavaScript is a "messy" language but god help me I love it.

Share

answered Nov 4, 2009 at 9:47

community wiki
Avi Y

2 I definitly have a Love/Hate relationship with JavaScript – Neil N Dec 9, 2009 at 17:04

+1, I know exactly what you mean. It can be fun to use. One thing I hate is the memory leaks. – JL. Apr 4, 2010 at 15:04

Aestehtically, it's a pile of dog-spew. Can't deny it gets the job done, though. – Engineer Sep 17, 2010 at 15:28

7 votes

Use unit tests as a last resort to verify code.

If you I want to verify that code is correct, I prefer the following techniques over unit testing:

1. Type checking
2. Assertions
3. Trivially verifiable code

For everything else, there's unit tests.

Share

answered Nov 8, 2009 at 17:55

community wiki
cdiggins

0. Re-read your code. Seems trivial, but often can be the best at finding errors. – Matt Hamsmith Nov 24, 2009 at 17:26

Enthusiasts of unit tests too often position their arguments as defenses for weak typing and late binding as if a disciplined engineer chooses exactly one approach to reliability.
– Integer Poet Mar 15, 2010 at 19:31

I'm very ambivalent about unit tests. My personal opinion is that zealots that want 100% code coverage for there unit tests are wasting a lot of time and money. But they're not completely useless either, so I guess I agree with the statement.
– Captain Sensible Apr 22, 2010 at 8:33

I've pretty much been forced to this conclusion by a very tight schedule. I agree that unit tests are not for everything. But having said that, the more critical a piece of code is, the wiser you'd be to write tests for it regardless. – Engineer Sep 17, 2010 at 15:26

7

votes

Not really programming, but I can't stand css only layouts just for the sake of it. It's counter productive, frustrating, and makes maintenance a nightmare of floats and margins where changing the position of a single element can throw the entire page out of whack.

It's definitely not a popular opinion, but i'm done with my table layout in 20 minutes while the css gurus spend hours tweaking line-height, margins, padding and floats just to do something as basic as vertically centering a paragraph.

Share

answered Nov 14, 2009 at 18:44

community wiki
Rob

1 Whoever spends hours writing `margin: 0 auto;` is one hell of a bad css-designer... Still, tables are tables and tables store data. Not design. – F.P Nov 18, 2009 at 14:18

1 That is why there are 3 different ways to use styles. For re-usability, and scope of need. – awright18 Mar 4, 2010 at 1:32

6

votes

I firmly believe that unmanaged code isn't worth the trouble. The extra maintainability expenses associated with hunting down memory leaks which even the best programmers introduce occasionally far outweigh the performance to be

gained from a language like C++. If Java, C#, etc. can't get the performance you need, buy more machines.

Share

answered Jan 2, 2009 at 13:27

community wiki
marcumka

2 if you can't track memory leaks, you're not worth to use high-powered tools. – Javier Jan 2, 2009 at 14:16

2 Not to mention that not all programs run exclusively on a recent version of Windows. – David Thornley Jan 2, 2009 at 14:54

2 I firmly believe that we don't need airplanes, we can always use cars, right...? And if we need to cross the open sea, we could just use a boat, right...? – Thomas Hansen Jan 10, 2009 at 20:54

1 Pipe-dream reasoning. Earth calling marcumka
– Captain Sensible Jan 26, 2009 at 10:41

7 **Right tool, right job.** Go try and code that kernel or NIC driver in C# and get back to us. Yes, there are plenty of folks who stick with the language they know, but your unqualified answer is overly broad. (And that from a Java developer!)
– Stu Thompson Apr 28, 2009 at 20:44

6 votes

Globals and/or Singletons are not inherently evil

I come from more of a sysadmin, shell, Perl (and my "real" programming), PHP type background; last year I was thrown into a Java development gig.

Singletons are evil. Globals are so evil they are not even allowed. Yet, Java has things like AOP, and now various "Dependency Injection" frameworks (we used Google Guice). AOP less so, but DI things for sure give you what? Globals. Uhh, thanks.

Share

answered Jan 2, 2009 at 17:32

community wiki
Jeff Warnica

I think you have some misconceptions about DI. You should watch Misko Hevery's Clean Code talks. – Craig P. Motlin Jan 2, 2009 at 18:37

I agree about globals. The problem is not the concept of a global itself, but what type of thing is made global. Used correctly, globals are very powerful. – Gene Roberts Jan 2, 2009 at 21:15

Perhaps I am. But if you had globals, you wouldn't need DI. I'm entirely prepared to believe that I'm mis-understanding a technology that solves a self-imposed problem. – Jeff Warnica Jan 4, 2009 at 5:24

We use Globals all the time in java, every time we use a final public static in place of a Constant (C, C++, C#). I think the thought is that if it needs to be global then it should be a static. I can (Mostly) agree with this. – WolfmanDragon Mar 30, 2009 at 19:21

6

votes

I think that using regions in C# is totally acceptable to collapse your code while in VS. Too many people try to say it hides your code and makes it hard to find things. But if you use them properly they can be very helpful to identify sections of code.

Share

answered Jan 2, 2009 at 19:25

community wiki
user18411

IMHO Regions are great for one thing... visualizing code rot. – Gavin Miller Jan 2, 2009 at 21:03

Never gotten used to them, don't use them, but it may just be me. – Captain Sensible Jan 26, 2009 at 15:38

Regions is the thing I miss most about VS (I use Eclipse). so instead of using regions, we make Method that have calls to methods that have calls to methods..... just so we can read the darned things. Regions are GOOD! +1 – WolfmanDragon Mar 30, 2009 at 19:17

6

votes

You shouldn't settle on the first way you find to code something that "works."

I really don't think this should be controversial, but it is. People see an example from elsewhere in the code, from online, or from some old "Teach yourself Advanced Power SQLJava#BeansServer in 3.14159 minutes" book dated

1999, and they think they know something and they copy it into their code. They don't walk through the example to find out what each line does. They don't think about the design of their program and see if there might be a more organized or more natural way to do the same thing. They don't make any attempt at keeping their skill sets up to date to learn that they are using ideas and methods deprecated in the last year of the previous millenium. They don't seem to have the experience to learn that what they're copying has created specific horrific maintenance burdens for programmers for years and that they can be avoided with a little more thought.

In fact, they don't even seem to recognize that there might be more than one way to do something.

I come from the Perl world, where one of the slogans is "There's More Than One Way To Do It." (TMTOWTDI) People who've taken a cursory look at Perl have written it off as "write-only" or "unreadable," largely because they've looked at crappy code written by people with the mindset I described above. Those people have given zero thought to design, maintainability, organization, reduction of duplication in code, coupling, cohesion, encapsulation, etc. They write crap. Those people exist programming in every language, and easy to learn languages with many ways to do things give them plenty of rope and guns to shoot and hang themselves with. Simultaneously.

But if you hang around the Perl world for longer than a cursory look, and watch what the long-timers in the

community are doing, you see a remarkable thing: the good Perl programmers spend some time seeking to find the **best** way to do something. When they're naming a new module, they ask around for suggestions and bounce their ideas off of people. They hand their code out to get looked at, critiqued, and modified. If they have to do something nasty, they encapsulate it in the smallest way possible in a module for use in a more organized way. Several implementations of the same idea might hang around for awhile, but they compete for mindshare and marketshare, and they compete by trying to do the best job, and a big part of that is by making themselves easily maintainable. Really good Perl programmers seem to **think hard** about what they are doing and looking for the best way to do things, rather than just grabbing the first idea that flits through their brain.

Today I program primarily in the Java world. I've seen some really good Java code, but I see a lot of junk as well, and I see more of the mindset I described at the beginning: people settle on the first ugly lump of code that seems to work, without understanding it, without thinking if there's a better way.

You will see both mindsets in every language. I'm not trying to impugn Java specifically. (Actually I really like it in some ways ... maybe that should be my real controversial opinion!) But I'm coming to believe that every programmer needs to spend a good couple of years with a TMTOWTDI-style language, because even though conventional wisdom has it that this leads to chaos and crappy code, it actually

seems to produce people who understand that you need to think about the repercussions of what you are doing instead of trusting your language to have been designed to make you do the right thing with no effort.

I do think you can err too far in the other direction: i.e., perfectionism that totally ignores your true needs and goals (often the true needs and goals of your business, which is usually profitability). But I don't think anyone can be a truly great programmer without learning to invest some greater-than-average effort in thinking about finding the best (or at least one of the best) way to code what they are doing.

Share

answered Jan 3, 2009 at 10:23

community wiki
skiphoppy

6

votes

Opinion: **There should not be any compiler warnings, only errors**. Or, formulated differently **You should always compile your code with -Werror**.

Reason: Either the compiler thinks it is something which should be corrected, in case it should be an error, or it is not necessary to fix, in which case the compiler should just shut up.

Share

answered Jan 3, 2009 at 17:39

community wiki
JesperE

I have to disagree. A really good warning system will warn you about things that are probably bad code, but may not be depending on how you use them. If you have lint set to full, I believe there are even cases where you can't get rid of all the warnings. – Bill K Jan 9, 2009 at 18:02

That would mean I would have to throw out my C# compiler. I have 2 (AFAIK, unfixable) warnings about environment references (that are indeed set correctly) that don't appear to break anything. Unless -Werror merely supresses warnings and doesn't turn them into errors >_> – user29053 Jan 9, 2009 at 20:53

Finally, someone disagrees. It wouldn't really be a controversial opinion otherwise, now would it? – JesperE Jan 9, 2009 at 22:35

Doesn't your C# compiler allow you to disable the warnings? If you know they are unfixable and "safe", why should the compiler keep warning? And yes, -Werror turns all warnings into errors. – JesperE Jan 9, 2009 at 22:37

I try to get the warnings down to zero but some warnings are 50:50: They make sense in case A but not in case B. So I end up sprinkling my code with "ignore warning"... :(– Aaron Digulla Feb 27, 2009 at 16:00

6
votes

VB sucks

While not terribly controversial in general, when you work in a VB house it is

Share

answered Jan 5, 2009 at 21:11

community wiki
gillonba

2 That this is not generally controversial shows how generally up themselves so many programmers are. Have a preference - fine. But when it comes down to whether you have a word (that you don't even have to type) or a '}' to terminate a block, it's just a style choice... – ChrisA Jan 7, 2009 at 14:15

... plenty of VB programmers suck, though. As do plenty of C# programmers. – ChrisA Jan 7, 2009 at 14:16

VB doesn't suck. People who use VB like VBA suck. – Chris Jan 8, 2009 at 3:30

VB *does* suck. So many things have been shoe-horned into what was originally a simple instructional language to allow novices to enter the domain of professionals that it's no longer appropriate for either novices nor professionals. – P Daddy Jan 9, 2009 at 13:57

It's not the language that sucks but a lot of the programmers that (used to) program in VB. – Captain Sensible Jan 26, 2009 at 10:39

6

votes

Two brains think better than one

I firmly believe that pair programming is the number one factor when it comes to increasing code quality and programming productivity. Unfortunatly it is also a highly controversial for management who believes that "more hands => more code => $$$!"

Share

answered Jan 9, 2009 at 13:40

community wiki
Martin Wickman

I sometimes dream about extreme extreme programming. How cool would it be if everyone in a group sat down to do the architecture and implementation as a group (4-8 devs). I wonder if it would work or be completely dysfunctional. I tend to think it could work with the "right" group. – oz10 Jan 14, 2009 at 3:18

6

votes

1. You should not follow web standards - all the time.

2. You don't need to comment your code.

As long as it's understandable by a stranger.

Share

answered Jan 10, 2009 at 0:08

community wiki
davethegr8

6

votes

As there are hundreds of answers to this mine will probably end up unread, but here's my pet peeve anyway.

If you're a programmer then you're most likely awful at Web Design/Development

This website is a phenomenal resource for programmers, but an absolutely awful place to come if you're looking for XHTML/CSS help. Even the good Web Developers here are handing out links to resources that were good in the 90's!

Sure, XHTML and CSS are simple to learn. However, you're not just learning a language! You're learning how to use it well, and very few designers and developers can do that, let alone programmers. It took me ages to become a capable designer and even longer to become a good developer. I could code in HTML from the age of 10 but that didn't mean I was good. Now I am a capable designer in programs like Photoshop and Illustrator, I am perfectly able to write a good website in Notepad and am able to write basic scripts in several languages. Not only that but I have a good nose for Search Engine Optimisation techniques and can easily tell you where the majority of people are going wrong (hint: get some good content!).

Also, this place is a terrible resource for advice on web standards. You should NOT just write code to work in the different browsers. You should ALWAYS follow the standard to future-proof your code. More often than not the fixes you use on your websites will break when the next browser update comes along. Not only that but the good browsers follow standards anyway. Finally, the reason IE was allowed

to ruin the Internet was because YOU allowed it by coding your websites for IE! If you're going to continue to do that for Firefox then we'll lose out yet again!

If you think that table-based layouts are as good, if not better than CSS layouts then you should not be allowed to talk on the subject, at least without me shooting you down first. Also, if you think W3Schools is the best resource to send someone to then you're just plain wrong.

If you're new to Web Design/Development don't bother with this place (it's full of programmers, not web developers). Go to a good Web Design/Development community like SitePoint.

Share

answered Jan 10, 2009 at 0:23

community wiki
Mike B

Goes for GUI design too. Especially with new technologies like WPF making GUI design more like web design with CSS like files defining styles for the interface. – Cameron MacFarland Jan 13, 2009 at 23:11

I completely agree, unfortunately, I find at most companies I'm the developer and the designer at the same time. Its like saying "hey, you're a good writer, you'd be a great illustrator too!" -- ummm, no. – Juliet Feb 4, 2009 at 0:35

6

votes

Relational Databases are a waste of time. Use object databases instead!

Relational database vendors try to fool us into believing that the only scaleable, persistent and safe storage in the world is relational databases. I am a certified DBA. Have you ever spent hours trying to optimize a query and had no idea what was going wrong? Relational databases don't let you make your own search paths when you need them. You give away much of the control over the speed of your app into the hands of people you've never met and they are not as smart as you think.

Sure, sometimes in a well-maintained database they come up with a quick answer for a complex query. But the price you pay for this is too high! You have to choose between writing raw SQL every time you want to read an entry of your data, which is dangerous. Or use an Object relational mapper which adds more complexity and things outside your control.

More importantly, you are actively forbidden from coming up with smart search algorithms, because every damn roundtrip to the database costs you around 11ms. It is too much. Imagine you know this super-graph algorithm which will answer a specific question, which might not even be expressible in SQL!, in due time. But even if your algorithm is linear, and interesting algorithms are not linear, forget about combining it with a relational database as enumerating a large table will take you hours!

Compare that with SandstoneDb, or Gemstone for Smalltalk! If you are into Java, give db4o a shot.

So, my advice is: Use an object-DB. Sure, they aren't perfect and some queries will be slower. But you will be surprised how many will be faster. Because loading the objects will not require all these strange transofmations between SQL and your domain data. And if you really need speed for a certain query, object databases have the query optimizer you should trust: your brain.

Share

edited Jan 12, 2009 at 8:53

community wiki
2 revs
nes1983

Wow that is controversial! Surprised you haven't been flamed by the other DBAs here ;) – Meff Jan 11, 2009 at 20:36

Even more important than performance: Development is much much faster with oo-databases! – wilth Apr 7, 2009 at 8:02

"Unskilled and Unaware of It: How Difficulties in Recognizing One's Own Incompetence Lead to Inflated Self-Assessments", Justin Kruger and David Dunning, Cornell University, Journal of Personality and Social Psychology, 1999, Vol. 77, No. 6., 121-1134. Fortunately it is curable (I'm the evidence): ".. Paradoxically, improving the skills of participants, and thus increasing their metacognitive competence, helped them recognize the limitations of their abilities." – MaD70 Nov 6, 2009 at 1:42

6

votes

You can't measure productivity by counting lines of code.

Everyone knows this, but for some reason the practice still persists!

Share

answered Jan 27, 2009 at 10:06

community wiki
Noel Walters

Do you realise the topic of the thread is "controversy"? How is your statement controversial? – Captain Sensible Jan 27, 2009 at 14:22

it depends who you're talking to. Metrics obsessed managers at my last job found it a very controversial point of view.
– Noel Walters Jan 27, 2009 at 18:05

6

votes

Reflection has no place in production code

Reflection breaks static analysis including refactoring tools and static type checking. Reflection also breaks the normal assumptions developers have about code. For example: adding a method to a class (that doesn't shadow some other method in the class) should never have any effect, but when reflection is being used, some other piece of code may "discover" the new method and decide to call it. Actually determining if such code exists is intractable.

I do think it's fine to use reflection and tests and in code generators.

Yes, this does mean that I try to avoid frameworks that use reflection. (it's too bad that Java lacks proper compile-time meta-programming support)

Share

answered May 14, 2009 at 15:18

community wiki
Laurence Gonsalves

Wouldn't this negate the possibility of developing an application that supports 3rd party plugins? – Steven Evers Jun 19, 2009 at 18:13

You're right, I should have been more clear. When I said "reflection" I meant java.lang.reflect. For plug-ins you just need Class.forName() and Class.newInstance(). I still consider the latter a "bad smell" (it's overused) but if you're implementing a system with third-party plugins then that's the way to do it. – Laurence Gonsalves Jun 20, 2009 at 1:14

6

votes

Garbage collection is overrated

Many people consider the introduction of garbage collection in Java one of the biggest improvements compared to C++. I consider the introduction to be very minor at best, well written C++ code does all the memory management at the proper places (with techniques like RAII), so there is no need for a garbage collector.

Share

answered Sep 16, 2009 at 10:45

community wiki
Anders Rune Jensen

The advocates of garbage collection have an unhealthy obsession with one particular resource when RAII covers all of them. – Integer Poet Mar 15, 2010 at 19:49

Lazy programmers suck. GC is for lazy programmers. Conclusion: you are totally right, Anders Rune Jensen. – user142019 Dec 20, 2010 at 14:58

6

votes

It's not the tools, it's you

Whenever developers try to do something new like doing UML diagrams, charts of any sort, project management they first look for the perfect tool to solve the problem. After endless searches finding not the right tool their motivation starves. All that is left then is complaints about the lack of useable software. It is the insight that the plan to be organized died in absence of a piece of software.

Well, it is only yourself dealing with organization. If you are used to organize you can do it with or without the aid of a software (and most do without). If you are not used to organize nobody can help you.

So "not having the right software" is just the simplest excuse for not being organized at all.

Share

edited Oct 31, 2009 at 17:39

community wiki
2 revs, 2 users 92%
Norbert Hartl

I think this is true in spite of people agreeing with it (figure that out). I make a pest of myself telling people that to do performance tuning you do not need a tool, in fact you may do better without one. – Mike Dunlavey Oct 31, 2009 at 17:42

6
votes

Don't be shy, throw an exception. Exceptions are a perfectly valid way to signal failure, and are much clearer than any return-code system. "Exceptional" has nothing to do with how often this can happen, and everything to do with what the class considers normal execution conditions. Throwing an exception when a division by zero occurs is just fine, regardless of how often the case can happen. If the problem is likely, guard your code so that the method doesn't get called with incorrect arguments.

Share

answered Nov 25, 2009 at 23:10

community wiki
Mathias

6 **Using regexs to parse HTML is, in many cases, fine**

votes

Every time someone posts a question on Stack Overflow asking how to achieve some HTML manuipulation with a regex, the first answer is "Regex is a insufficient tool to parse HTML so don't do it". If the questioner was trying to build a web browser, this would be a helpful answer. However, usually the questioner wants to do some thing like add a rel tag to all the links to a certain domain, usually in a case when certain assumptions can be made about the style of the incoming markup, something that is entiely reasonable to do with a regex.

Share

answered Dec 6, 2009 at 14:06

community wiki
Nick Higgs

6

votes

According to the amount of feedback I've gotten, my most controversial opinion, apparently, is that programmers don't always read the books they claim to have read. This is followed closely by my opinion that a programmer with a formal education is better than the same programmer who is self-taught (but not necessarily better than *a different* programmer who is self-taught).

Share

edited Jan 6, 2023 at 8:55

community wiki

3 revs, 2 users 82%
Bill the Lizard

I'm proud to say I've read all the programming books I own. Even the monsterous Programming Python and Programming Perl. – user13876 Jan 5, 2009 at 11:42

I have a B.A. in English. It is likely that I'm a better programmer for it. Is that what you mean? – postfuturist Jan 10, 2009 at 0:41

Not exactly. – Bill the Lizard Jan 22, 2009 at 14:26

You over-estimate the value of education. I've been a full time programmer for 15 years and am self-taught. When I meet developers who are fresh out of school, I sometimes wonder if there whole education wasn't a big waste of time. They know next to nothing about "the real world", can seldomly work independently and their skills are average at best.
– Captain Sensible Apr 22, 2010 at 8:43

@Seventh Element: I would expect someone fresh out of school with no work experience to have average skills. Comparing a fresh graduate to someone with 15 years of work experience is comparing apples to oranges. I worked as a programmer for 8 years before going back to school to get my degree. I think I have a pretty strong grasp of the value of my education *to me*. You get out of it what you put into it.
– Bill the Lizard Apr 22, 2010 at 12:14

5
votes

In my workplace, I've been trying to introduce more Agile/XP development habits. Continuous Design is the one I've felt most resistance on so far. Maybe I shouldn't have phrased it as "let's round up all of the architecture team and shoot them"... ;)

Share

answered Jan 2, 2009 at 13:20

community wiki
Giraffe

> That's good. Along the same lines is casually insulting people in the name of "truth". That particular virus seems to have a reservoir in grad schools, like the one I attended.
> – Mike Dunlavey Nov 3, 2009 at 14:38

5
votes

Opinion: Data driven design puts the cart before the horse. It should be eliminated from our thinking forthwith.

The vast majority of software isn't about the data, it's about the business problem we're trying to solve for our customers. It's about a *problem domain*, which involves objects, rules, flows, cases, and relationships.

When we start our design with the data, and model the rest of the system after the data and the relationships between the data (tables, foreign keys, and x-to-x relationships), we constrain the entire application to how the data is stored in and retrieved from the database. Further, we expose the database architecture to the software.

The database schema is an implementation detail. We should be free to change it without having to significantly alter the design of our software at all. The business layer should never have to know how the tables are set up, or if it's pulling from a view or a table, or getting the table from

dynamic SQL or a stored procedure. And that type of code should ***never*** appear in the presentation layer.

Software is about solving business problems. We deal with users, cars, accounts, balances, averages, summaries, transfers, animals, messsages, packages, carts, orders, and all sorts of other real tangible objects, and the actions we can perform on them. We need to *save*, *load*, *update*, *find*, and *delete* those items as needed. Sometimes, we have to do those things in special ways.

But there's no real compelling reason that we should take the work that should be done in the database and move it away from the data and put it in the source code, potentially on a separate machine (introducing network traffic and degrading performance). Doing so means turning our backs on the decades of work that has already been done to improve the performance of stored procedures and functions built into databases. The argument that stored procedures introduce "yet another API" to be manged is specious: of course it does; that API is a facade that shields you from the database schema, including the intricate details of primary and foreign keys, transactions, cursors, and so on, and it prevents you from having to splice SQL together in your source code.

Put the horse back in front of the cart. Think about the problem domain, and design the solution around it. Then, derive the data from the problem domain.

answered Jan 2, 2009 at 17:46

community wiki
Mike Hofer

1 I agree with the principal, but the problem is in real world IT development you often have existing data stores that you must make use of - while total constraint to existing code might be bad you can save a ton of development effort if you conform to data standards that exist when you can.
– Kendall Helmstetter Gelner Jan 5, 2009 at 5:28

Hey, someone who understands the real purpose of stored procedures! – Lurker Indeed Jan 12, 2009 at 18:04

2 Hmmm. Take the data out of a system and what do you have? A system that computes nothing. Put bad data into your system and what happens? Crash. Analogy: Bake your bricks (create strong data types) and mix your cement (enforce the constraints), then design/build your system with perfect blocks.
– Triynko Apr 17, 2009 at 2:53