Community

yegor256.com

22 Comments • Created a year ago

Typical Mistakes in Java Code

Typical Mistakes in Java Code

This page contains most typical mistakes I see in the Java code of people working with me. Static analysis (we're using qulice can't catch all of the mistakes for obvious reasons, and that's why I decided to list them all here. Let me know if y…

(yegor256.com)

22 Comments

Recommend Share Sort by Newest

Join the discussion…

Dmitri Pisarenko • 5 months ago

The link "USPS abbreviates street names" (https://www.usps.com/send/offi...) is no longer valid. Please update it.

• Reply • Share ›

Yegor Bugayenko author ➚ Dmitri Pisarenko • 5 months ago

Thanks, the right link is this: http://pe.usps.gov/text/pub28/... I will update the article soon.

• Reply • Share ›

pandiaraj • 10 months ago

IMHO, variable names should be composite. In case of HttpRequest request, developer may get confused about the type of request object few lines down the code. Also, if that class has a FtpRequest field, how will you name it? Similarly, File content? What if there are couple of files need to be accessed? It would be better if we name the variables like File userFile, File logoFile, etc, it would increase the readability of the code. And File object is not the content, it represents the 'File', not the content. If I read a variable named 'content', I would assume it is file content and thus it should be String or Byte[] type :)

Would like to know your suggestions.

• Reply • Share ›

Yegor Bugayenko author ↗ pandiaraj • 10 months ago

That's the idea - if you need to give complex names to your variables, your design has flaws. It is too complex. If you need to name two variables in the same scope (class or method) `logoFile` and `userFile`, you need to think about making your scope smaller.

Another point you mentioned, about file vs. content. It is against the entire OOP paradigm - knowing the type of your variable. You should NOT know or make any assumptions about variable types, when using it. You should rely on the behavior it exposes. If you have content, call it `content`, either it is a file, or an array of bytes, or a string. Then, ask it to give you, for example, the size of itself. It will give you an answer. That's it. Simply put, object thinking vs. procedural thinking :)

1 ▲ | ▼ • Reply • Share ›

pandiaraj ↗ pandiaraj • 10 months ago

Forgot to mention, excellent article :)

▲ | ▼ • Reply • Share ›

Vach • a year ago

I think you are a perfectionist (just like me) and thats a good thing unless you have a deadline and a night to develop lot of stuff :D

▲ | ▼ • Reply • Share ›

Yegor Bugayenko author ↗ Vach • a year ago

I believe that true perfectionists don't have overnight deadlines :) They build everything perfectly, including their plans :)

2 ▲ | ▼ • Reply • Share ›

Vach ↗ Yegor Bugayenko • a year ago

Well when you did acquire enough experience to write some code that is close to "perfect code" you wont have for sure, but while you are learning you always have that wish to do things the right way but not necessarily enough time to do so...

▲ | ▼ • Reply • Share ›

Allan Clarke ↗ Vach • 8 months ago

Being a perfectionist is better than being a slob when it comes to programming, we all would probably agree. Nobody wants their code base to die the death of a thousand cuts. On the flip side, no business is going to pay you to keep polishing code until you feel it is perfect (of course). So your job, as a craftsman, is to balance cost and product. (Sorry to state the obvious).

▲ | ▼ • Reply • Share ›

biqo • a year ago

What about the name Iterator ,it ends in r?

1 • Reply • Share ›

Yegor Bugayenko author ➦ biqo • a year ago

It's a bad name indeed. `Stream` or a `Items` would sound much better, check this article: http://www.yegor256.com/2014/1...

• Reply • Share ›

carfield • a year ago

And, of course, utility classes are anti-patterns, like StringUtils, FileUtils, and IOUtils from Apache. The above are perfect examples of terrible designs. <- Just wonder, how should apache team name those classes?

• Reply • Share ›

Yegor Bugayenko author ➦ carfield • a year ago

How about ApacheString and ApacheFile? Instead of StringUtils.strip(name) we would do new ApacheString(name).strip().

1 • Reply • Share ›

Willem Salembier ➦ Yegor Bugayenko • 8 months ago

I don't get this blind obsession of writing every line of code in a strict OOP style. There is nothing wrong with having some static helper methods doing low-level operations.

- What real-life entity does an ApacheString represent? A self-stripping character sequence?
- Why should I instantiate and garbage collect an object for that?

I'm not convinced. I think it leads to verbose and non-performant code and that you try to apply OOP on a much too low-level scale.

1 • Reply • Share ›

Yegor Bugayenko author ➦ Willem Salembier • 8 months ago

A better example would be to have `StrippedString`, which would extend `String`, and we would use it like this:

```
CharSequence text = new Capitalized(new Stripped(" hello, world! "
System.out.println(text); // "Hello, World!"
```

This code is not more verbose that this (it is less verbose, actually):

```
String text = StringUtils.capitalize(StringUtils.strip(" hello, wo
System.out.println(text); // "Hello, World!"
```

Instantiation and garbage collection is a valid point. It is a problem of JVM, not of OOP. I will agree with you if you say "I want to use objects, but since my application requires highest performance and I'm writing in Java for Oracle JDK, I have to use static methods". That is a valid excuse. But just an excuse and exception, not a rule

That is a valid excuse. But just an excuse and exception, not a rule. As a common rule we should use objects, when we're in OOP. See my point?

Look at this article also: http://www.yegor256.com/2014/0...

▲ | ▼ • Reply • Share ›

Oliver Doepner ➚ Yegor Bugayenko • 4 months ago

1) Nothing can extend String, because java.util.String is a final class.
2) Verbosity of static method calls is much reduced by static imports, e.g. capitalize(strip(" hello, world! ")).
3) With all your constructor calls you basically also use static methods (constructors are essentially static methods), which makes your code depend on all those concrete classes. I would use non-static methods in an immutable StringHelper class that implements an interface, for example called "Strings". The helper instance can be created as a de-facto singleton (not a GoF singleton) early in your application and be injected into the classes that need it. No object creation overhead, flexible and DI-friendly.
4) I generally try to avoid static util classes and methods. But for short methods that only have one optimal, correct implementation I still use static utils - like stateless mixins via static imports - like in this example:
public static boolean isNullOrEmpty(String s) { return s == null || s.isEmpty(); }

▲ | ▼ • Reply • Share ›

Andrej Istomin ➚ Yegor Bugayenko • 7 months ago

Reading this discussion I remember my experience with Android. I know you will be happy reading this: http://developer.android.com/t.... I'm quoting: "Avoid creating unnecessary objects". Google made his own Java with Activities and without OOP :) BTW, I never saw such a recommendations regarding iOS(I note it, because somebody will say that mobile development and OOP are incompatible).

▲ | ▼ • Reply • Share ›

Yegor Bugayenko author ➚ Andrej Istomin • 7 months ago

Android SDK is a classic example of a horrible design. Look at that Activity class for example: http://developer.android.com/r... 140 public methods. Huh? It is object-oriented programming? And it's coming from Google. The entire industry is in trouble :(

2 ▲ | ▼ • Reply • Share ›

Andrej Istomin ➚ Yegor Bugayenko • 7 months ago

When I was working with their SDK, sometimes I had and idea, that, probably, they just trolling the engineers society :) There is a humor even in names, like WTF method of logger: http://developer.android.com/r..., java.lang.String)

By the way, about activities... how to add dependency to activity? :) It was really painful to work with such tools.

• Reply • Share ›

Guest Andrej Istomin • 7 months ago

By the way, about activities... how to add dependency to activity? :) It was really painful to work with such tools.

• Reply • Share ›

carfield Yegor Bugayenko • a year ago

I see, thanks for sharing.

• Reply • Share ›

More from yegor256.com

yegor256.com recently published

Three Things I Expect From a Software Architect

23 Comments Recommend

yegor256.com recently published

How to Avoid a Software Outsourcing Disaster

8 Comments Recommend

yegor256.com recently published

Constructors Must Be Code-Free

68 Comments Recommend