



Community

 yegor256.com

62 Comments · Created 5 months ago



## A Compound Name Is a Code Smell

Do you name variables like `textLength`, `table_name`, or `current-user-email`? All three are compound names that consist of more than one word. Even though they look more descriptive than `name`, `length`, or `email`, I would strongly recom...

[\(yegor256.com\)](#)

62 Comments

 Recommend Share

Sort by Newest ▾



Join the discussion...

**coder.slynk** · 3 months ago`firstName` -> `first`? First what?`lastName` -> `last`? Last what?`createdAt` -> `created`? Is it now a boolean? I can't tell.

Clinging on to arbitrary rules instead of working to make readable/maintainable code is the sign of a bad software developer.

1 ^ | v · Reply · Share ›

**John Doe** · 3 months ago

Your article is about names, but in your example you are providing just field/variable example. What about compound class and method names?

^ | v · Reply · Share ›

**Oleg Majewski** → John Doe · 3 months ago

try this rule for classes and methods:



try this rule for classes and methods.

big scope, short name: `new File(".").read()`,

small scope, long name: `private byte[] readBytesFrom(String path)`

^ | v · Reply · Share ›



**Yegor Bugayenko** author → John Doe · 3 months ago

Yes, I'm talking mostly about variable names here. Class names is a different story. Method names is yet another story. I'll write about them later, I guess.

^ | v · Reply · Share ›



**Marcos Douglas Santos** · 3 months ago

What about class names, why do you use suffix like "Rs" <https://github.com/yegor256/ru...> or prefix like "Dy" <https://github.com/yegor256/ru...> ? Are better names?

I like to use a "prefix of context" for all classes. Context could be an acronym to an module, sector in the company, so on.

Eg:

Customer, Order interfaces

ABCustomer, ABOrder classes... belongs the sector AB of a company.

FnCustomer, FnOrder classes... belongs to another sector (finances) of this company.

Why that? Well we need only one interface for Customer and Order, but the context is different for each sector/module so, different implementations. Do you agree?

What the differences between my method and yours? Your prefix/suffix looks like that exists only to not have name collision between interface and class. My prefix exists to define a context.

^ | v · Reply · Share ›



**Oleg Majewski** → Marcos Douglas Santos · 3 months ago

hi, your example is good example. Class names should be in general short, then if you derive it gets longer.

your example Order and FnOrder is good example for that rule ;-) I think FinancialOrder would be also good.

1 ^ | v · Reply · Share ›



**Yegor Bugayenko** author → Marcos Douglas Santos · 3 months ago

I'm using these two-letter prefixes for two reasons - they are compact and groupable. Instead of "TextDocument" and "BinaryDocument" I'm using "DtText" and "DtBinary". They are sortable and look much nicer.

^ | v · Reply · Share ›



**Marcos Douglas Santos** → Yegor Bugayenko · 3 months ago

But choose compact prefix only because it is, compact, makes class names less readable, don't?

^ | v · Reply · Share ›



**Yegor Bugayenko** author → Marcos Douglas Santos · 3 months ago

I think it's a matter of taste. As long as naming is consistent, it is



I think it's a matter of taste. As long as naming is consistent - it is readable (and understandable).

^ | v · Reply · Share ›



**Oleg Majewski** · 3 months ago

What about:

```
/**
 * some data object
 */
public class MyData {
    // different ids example
    private final Long customerId;
    private final Long dealerId;
    private final Long orderId;
    // different customers example
    private final Customer commercialCustomer;
    private final Customer simpleCustomer;
}
```

^ | v · Reply · Share ›



**Yegor Bugayenko** author → Oleg Majewski · 3 months ago

This class is begging for refactoring, and compound names are perfect indicators of the problem.

^ | v · Reply · Share ›



**Oleg Majewski** → Yegor Bugayenko · 3 months ago

how would you refactor it? why?

think about message data object / special data tables like star schema etc., there is no (business) logic in that class.

^ | v · Reply · Share ›



**Yegor Bugayenko** author → Oleg Majewski · 3 months ago

First of all, a class without any logic is a terrible design. It's a data structure, not a class. Second, the name of the class is terrible, since it doesn't explain what the class is, but labels it as a data bag. Check this article, it explains this idea in more details:

<http://www.yegor256.com/2014/1...>

^ | v · Reply · Share ›



**Oleg Majewski** → Yegor Bugayenko · 3 months ago

fair enough, it was an useless example to keep it simple and use case neutral. Lets make another more useful example:

```
/**
```

```

* Represents a message to be send if a price change occurs
*/
public class PriceChange { // without interface, is not the focus
    private final Money from;
    private final Money to;
    private final Long articleId; // needed by consumers to retrieve
    private final Long customerId; // needed by consumers to retri

    // omitted ctor && getters to keep it simple

    public void send() {
        // messaging system logic
    }
}

```

^ | v • Reply • Share ›



**Yegor Bugayenko** author → Oleg Majewski • 3 months ago

Use `article` and `customer` instead. Why do you need that "id" suffix?  
You have one article and one customer in scope, why the suffix?

^ | v • Reply • Share ›



**Oleg Majewski** → Yegor Bugayenko • 3 months ago

expected this :)

well, you already know the answer, because it would be inconsistent  
as everywhere else in the source base `customer` stands for:

```

private final Customer customer; // the full object, not only id

```

But for the message the id is enough, the suffix makes it clear to the reader of the code.

To complete the example, a typical consumer would do something like:

```

public void consume(PriceChange message) { // used message instead
    // same issue here, but with repos
    Article article = articleRepository.findById(message.getAr
    Customer customer = customerRepository.findById(message.ge

    // Lets try how it would looks like if the names would be
    // Article article = articleRepository.findById(message.ge
    // Customer customer = customerRepository.findById(message
    // Looks ugly?

    // do something useful on price change
}

```

^ | v · Reply · Share ›



**Yegor Bugayenko** author → Oleg Majewski · 3 months ago

This is the code you're used to, but it doesn't mean it's a good code. These "getArticleId()" methods are a perfect sign of a bad design. Check this article: <http://www.yegor256.com/2014/0...>

^ | v · Reply · Share ›



**Oleg Majewski** → Yegor Bugayenko · 3 months ago

>This is the code you're used to, but it doesn't mean it's a good code

no it does not, never said, it is an example, it could be wrong, but you are not convincing enough.

>These "getArticleId()" methods are a perfect sign of a bad design.

What would be a good design for that message in this use case in your opinion? Your answer sounds like Consumer/Producer pattern is a bad design, is it really your opinion?

it's not enough to say it is bad and check your own article with your use case which is perfect for that article.

Check your article: <http://www.yegor256.com/2015/0...> "No Bullshit"

1 ^ | v · Reply · Share ›



**Marcos Douglas Santos** → Oleg Majewski · 3 months ago

It is a bad design because you have -- Edit:~~only~~ -- a data structure, that have three distinct information -- Money (from/to), articleId and customerId -- that belongs to three others objects. Using compound names, as **@Yegor Bugayenko** said before, are perfect indicators of the problem.

Looking just for this example, the code is "wrong" -- if you considered OOP, immutable objects, so on -- but could make sense in a bigger context, for example to performance.

^ | v · Reply · Share ›



**Oleg Majewski** → Marcos Douglas Santos · 3 months ago

no it's not only data, please read the examples more careful. Just to say "don't use compound names" and then say everything else is wrong, is not enough, this is what the example and question is about :)

^ | v · Reply · Share ›



**Marcos Douglas Santos** → Oleg Majewski · 3 months ago

I changed my answer above :)

Ok, you have a send() method, so what? IMHO you continues using three attributes on this class -- Money (from/to), articleId and customerId -- that belongs to three others distinct classes. That is the

customerId -- that belongs to three others distinct classes. That is the problem.

^ | v · Reply · Share ›



**Oleg Majewski** → Marcos Douglas Santos · 3 months ago

sure, I fully understand that it is "wrong" by definition of yegor

>that belongs to three others distinct classes

you are making it to easy for you, btw. there are of course also a Customer and Article classes with that ids and more.

^ | v · Reply · Share ›



**Marcos Douglas Santos** → Oleg Majewski · 3 months ago

sure, I fully understand that it is "wrong" by definition of yegor

Well, if you are using OOP this is, really, wrong. Your class have a method with logic, but the attributes belongs to others classes.

you are making it to easy for you, btw. there are of course also a Customer and Article classes with that ids and more.

Ok, you have more classes but I'm talking about PriceChange class that was the example you posted.

^ | v · Reply · Share ›



**Oleg Majewski** → Marcos Douglas Santos · 3 months ago

>Well, if you are using OOP this is, really, wrong.

again, it's not enough to say it is wrong ;-)) what would be "right" and solves the same problem?

>Your class have a method with logic, but the attributes belongs to others classes.

no the attributes is that what I want to send within the message using a messaging system. Thus they belong to that class. I don't want to send the entire objects, only the ids :) I also want a java class (PriceChange) to access the data was send within that message (inside PriceChange itself)

^ | v · Reply · Share ›



**Marcos Douglas Santos** → Oleg Majewski · 3 months ago

The "right" to solves this "problem" I've already said... but again, in another words: you do not have a "real" object. You have "data structure" that have a misc of attributes. The motive I don't know. Maybe performance, simplicity, whatever.

■ . . . . .

I don't want to send the entire objects, only the ids

So you WANT to do like that. I don't know what to say more. :)

^ | v · Reply · Share ›



**Oleg Majewski** → Marcos Douglas Santos · 3 months ago

there is nothing to be said any more, it would be helpful to provide a source code which solves the same problem, but is done in the "right" way ;-) To repeat the problem: you have a producer and a consumer, the producer generates the PriceChange event (message), consumer needs to react on it and do something useful. To do something helpful it needs to fetch consumer and article by id first from there own datastores. (we are still talking about compound names ;-))

^ | v · Reply · Share ›



**Marcos Douglas Santos** → Oleg Majewski · 3 months ago

We are talking about theoretical object-oriented and what is "right" following the pure OOP concepts or we talking about how to improve your design code specifically?

I'm talking about the first one. As I said before, you chose this design for some reasons... but when I said "wrong" I meant: this design not following the concepts of OOP.

^ | v · Reply · Share ›



**Oleg Majewski** → Marcos Douglas Santos · 3 months ago

we are talking about an example which is OOP (btw. it has nothing to do with that huge restrictions from yegor, which goes much more far away then restrictions of OOP itself) where that yegors restriction can't work/ trying to find an example where it does not make sense and to illustrate it.

^ | v · Reply · Share ›



**Yegor Bugayenko** author → Oleg Majewski · 3 months ago

Oleg, your code has many different issues, but let's try to focus on one of them - the name "articleId". Why do you need this "Id" suffix? What information it gives you? Your scope is rather small and using just "article" won't make it unclear. I can tell you why you're using it. Because this is a tradition, a habit, a common practice. Where did it come from? From bad programmers who are using dozens of variables in the same scope. In that case we have to use that suffices. Otherwise we won't be able to distinguish an "article" from "articleId" and "articleTitle" and "articleCanHaveComments". That bad programmers are teaching us, for years, that such naming is the only possible mechanism to differentiate article from its ID. But in this article I'm saying that it's not true. It's not the only mechanism. There is another approach - refactoring and decomposing a big



unmaintainable scope into pieces. See my point?

1 ^ | v · Reply · Share ›



**Oleg Majewski** → Yegor Bugayenko · 3 months ago

found one more example, hopefully better:

```
private PresentableCodecast formatCodecast(User loggedInUser, Code
PresentableCodecast cc = new PresentableCodecast();
cc.title = codecast.getTitle();
cc.publicationDate = dateFormat.format(codecast.getPublicationDa
cc.isViewable = isLicensedFor(VIEWING, loggedInUser, codecast);
cc.isDownloadable = isLicensedFor(DOWNLOADING, loggedInUser, cod
return cc;
}
```

^ | v · Reply · Share ›



**Yegor Bugayenko** author → Oleg Majewski · 3 months ago

It's a good example of a bad code. Take, for example, this "loggedInUser" variable. Why can't we replace it with "user"? Do we have many users in this method? Only one. So why we're giving it such a long name? This is the same as naming those well known characters Adam Godsend Sr. and Eve Maria Godsend. Sounds ridiculous isn't it? That's how ridiculous this code looks to me :)

^ | v · Reply · Share ›



**Oleg Majewski** → Yegor Bugayenko · 3 months ago

If you would know where this code comes from you would think twice before you brand mark it as bad (hint it was not written by me), but here the last try to help you out:

this is how you READ this code:

to format a code cast for a logged in user and a code cast

you need to create an empty presentable codecast first

then you set the title of the codecast you want to present

you also need to format the publication date of the codecast

then you need to decide if your codecast is visible if it is licensed for viewing for that particular logged in user and the codecast

after all you need to decide if it can be downloaded if it is licensed for downloading for that logged in user and the codecast

maybe my description of that code is not ideal, but if you still don't get the point, it was the last try.

^ | v · Reply · Share ›



**Oleg Majewski** → Yegor Bugayenko · 3 months ago

I let's answer the last question first



>There is another approach - refactoring and decomposing a big unmaintainable scope into pieces. See my point?

Totally with you! But here the article is just about compound names, not about decomposing. I tried to find an example where this rule is too strict and contra productive, seems like I failed with my example (twice ;-)). But I still think, that what you try to do is to enforce decomposing, not only by this rule, also by rules like empty line etc. But at the end of the day you know there are situations where it is useless and even disruptive.

>Oleg, your code has many different issues, but let's try to focus on one of them - the name "articleId". Why do you need this "Id" suffix? What information it gives you? Your scope is rather small and using just "article" won't make it unclear

Let's answer that question first in this way: I have been already working with code, where something called "content" was used to save "groups", and the reason for that was: "because it has the same behavior and easy to do, but renaming would be hard, because we are using content already for so many other different things". So after some time working with the code I knew that "content" stands for group, in the scope of that piece of code, I was working with. In another scope it was standing clearly for something else, and of course it was clear to every one, who worked with the code already for a longer time, and if you don't understand that smart reusage of the same thing, then you have to learn it (understand the scope). You see my point? ;-)

Ok now without fun, I already answered this question: "But for the message the id is enough, the suffix makes it clear to the reader of the code" -> the suffix "Id" makes clear to the reader of the code, that the value is a simple primitive and not a, let's call it, struct.

in code:

articleId stands for: { 42 }

article stands for { "id": 42, "name": "batman", "description": "very black" }

again, as the reader of the code it is very clear to everyone what an articleId stands for, but if you write just article, then you first need to understand the scope around it, this is not what clean code teaches you ;-)

Now really, a 42 is definitely not an article!

^ | v · Reply · Share >



**dbbohlín** · 5 months ago

I agree that bad variable names is a "code smell", just not in the way that you suggest. Having written a lot of code and more importantly, in this context at least, maintained a lot of

code. I can tell you that proper variable names is very important to the readability of the code.

You show variable names that are bad and are ugly but you don't show an example of a good variable name. I think it is a good thing to talk about variable names and at least it isn't Polish Notation such as `strFile`, which while a reasonably good idea at the time very quickly outlived its usefulness.

But a variable name needs to very QUICKLY give GOOD INFORMATION to the person reading the code so they don't have to continually read back or make notation on a variable name just to move forward in the code. There is nothing more frustrating than when you are reading someone else's code and they don't have comments and they don't have good variable names and they are no longer there to tell you "Oh, yea, I named it that because it was a reminder of my partner who's name rhymes with file name."

You start out with a class called `CSV`. How is that helpful? `CSV` what? `CSV` formatter? .. parser? .. writer? What is the class doing other then `CSV` or is it doing every thing `CSV`? (I could even get a little more pedantic and say what, `CSV` Comma Separated Values, Charlie Samuel Voltimort? but really `CSV` is common enough that most people would understand it).

Then you names your variable `csvFileName`. That isn't good either, but not for the reason you say. It is not good because it gives too much information. Does the method care that it is a `CSV` file name? Well, no it doesn't, it cares that it is a file name but it is inferred that it is related to `CSV` because it is in a class called `CSV`.

So, you then refactor it into `"file"`, which is just as bad because now I am really confused. Is it a `File` Object? Is it a file name? Is it a file path? What part or all of a file are we talking about? Well it isn't a file because it doesn't have all of the attributes of a file, it only has the name attribute of a file. Even then, there is ambiguity, because we don't know if what is required is the file's name or the file's fully qualified path? If it just the name then fine `fileName` is a good variable name. Why? Because I, as a stranger to the code, can instantly KNOW what is required for this method to work. If it is the file path then the name should be `filePath` so that I don't have to guess what required.

Think of it this way. If I am in a crowded room and I call out "Hey! Person!" I am going to get a bunch of confused looks. If I call out "Hey! John!" I probably will only confuse all of the 'Johns' in the place. If I call out "Hey! John Smith!" I probably will get what I want, and if I need to be absolutely sure I should probably call out "Hey! John Joseph Smith!" so that I am absolutely clear to everyone what I am talking about.

Variable names aren't for your sake, they are for the people who come later. As you write your code, think about the poor sucker who, after you have long gone, has to come in a make sense of that 4,000,000 line program you wrote and no longer have to maintain.

9 ^ | v • Reply • Share ›



**Oleg Majewski** → dbbohlín • 3 months ago

>But a variable name needs to very QUICKLY give GOOD INFORMATION

+1

^ | v · Reply · Share ›



**Jacob Zimmerman** · 5 months ago

Just using file as the name is misleading. A user might think that it's asking for some sort of File object instead of just a string of the name of the file. And "name" would be ambiguous as well.

By the way, you didn't actually show how this is a code smell. Code smell implies that there is some sort of bad design that will be improved with change (and simply renaming a few variables isn't really an improvement). You also didn't show how the change improved anything.

Lastly, Eve was named by Adam, not "you know who".

5 ^ | v · Reply · Share ›



**Yegor Bugayenko** author → **Jacob Zimmerman** · 5 months ago

Once you see a long name, it's time to refactor the code, because it's too big. That's what the "smell" part is about. Once you hear the smell, do the refactoring and break the scope down into smaller scopes. Maybe a better/longer example would convey the idea better, I agree.

You're right about Eve, my fault :) Thanks for correcting!

^ | v · Reply · Share ›



**Jacob Zimmerman** → **Yegor Bugayenko** · 5 months ago

You'd have to provide a much better example, because you didn't break any scopes down.

^ | v · Reply · Share ›



**Yegor Bugayenko** author → **Jacob Zimmerman** · 5 months ago

Here it is. Suppose you have a method that copies one file to another:

```
void copy(String sourceFileName, String destFileName) {
    File sourceFile = new File(sourceFileName);
    File destFile = new File(destFileName);
    try (InputStream sourceStream = new FileInputStream(sourceFile);
        OutputStream destStream = new FileOutputStream(destFile)) {
        while (true) {
            int data = sourceStream.read();
            if (data < 0) {
                break;
            }
            destStream.write(data);
        }
    }
}
```

We have compound names here. If we want to make them shorter and use only nouns - we can't do it without refactoring. We can't rename sourceFile to source, because there is also sourceStream. In

order to use nouns only we will have to break this scope down to smaller parts:

```
void copy(String source, String dest) {
    copy(new File(source), new File(dest));
}

void copy(File source, File dest) {
    try (InputStream input = new FileInputStream(source);
        OutputStream output = new FileOutputStream(dest)) {
        copy(input, output);
    }
}

void copy(InputStream source, OutputStream dest) {
    while (true) {
        int data = source.read();
        if (data < 0) {
            break;
        }
        dest.write(data);
    }
}
```

Looks better now, isn't it? :)

2 ^ | v · Reply · Share ›



**Leon** → Yegor Bugayenko · 3 months ago

What's a "dest"? Seriously, name abbreviations are hacker mentality and a code smell on their own...

Also, "String source" doesn't really say anything, since String is a multipurpose type. How are we supposed to know it takes two file names, just by looking at the signature?

1 ^ | v · Reply · Share ›



**ricardojlrufino** → Leon · a month ago

maybe that copy the contents of a string to the other .. =]

^ | v · Reply · Share ›



**Jacob Zimmerman** → Yegor Bugayenko · 5 months ago

This is much better. I actually knew what you meant from the beginning, but I had to argue for a better example because the first wasn't really an example and because I needed to make sure you realized it was, as you say, a smell, which isn't necessarily a bad thing, but gives hints that there could be a better solution.

Thanks for being a good sport.

^ | v · Reply · Share ›



**cfoley** → Yegor Bugayenko · 5 months ago



I like this example and it's the kind of method overloading I like to do in Java. However, given that your original example was in Ruby, how would you refactor a similarly structured piece of code in Ruby? You cannot use method overloading in the same way without using compound method names.

^ | v · Reply · Share ›



**Yegor Bugayenko** author → cfoley · 5 months ago

That's true, Java is more convenient due to its strict typing. However, in Ruby you can give different names to the methods, like "copyFiles", "copyStreams", etc. Or even create small supplementary classes and then call copy() method on them, like "Files#copy()", "Streams#copy()", etc.

^ | v · Reply · Share ›



**cfoley** → Yegor Bugayenko · 5 months ago

Files#copy(), etc. means making more classes to represent similar things that exist in Ruby. You'll want to do that sometimes but doing it dogmatically will lead to a confusing codebase.

Ruby does give you the option to open classes up so you could put your new methods inside existing classes. However, opening up existing classes is widely discouraged, and for good reason.

This leaves method names like "copyFiles", "copyStreams", etc. It certainly seems like the pragmatic solution but it's at odds with your dogmatic stance on compound names. All you have done is shift the compound names from variable names to method names.

Three methods to copy things could be seen as either a more flexible design or as unnecessary clutter depending on the codebase, but it looks like you have to put those compound names somewhere and where they go seems like the least important factor in making a design decision.

^ | v · Reply · Share ›



**Джек** · 5 months ago

Russians just like to tell everybody how bad everything around is. :-P

1 ^ | v · Reply · Share ›



**Alex** · 5 months ago

I'm confused, does CSV take a file (a binary stream) or a file name or both?

1 ^ | v · Reply · Share ›



**Yegor Bugayenko** author → Alex · 5 months ago

That's indeed a problem with Ruby, that is not strictly typed. But in Java it's much more clear, since the method declaration would look like this:

```
class CSV {
  public CSV(File source) {
  }
}
```

1 ^ | v • Reply • Share ›



**Marcos Douglas Santos** → Alex • 5 months ago

Because Ruby has no type declaration in the arguments, creates this confusion :)

1 ^ | v • Reply • Share ›



**Alex** → Marcos Douglas Santos • 5 months ago

I was just being flippant. I wanted to convey that there is indeed ambiguity and confusion since "file" does not convey the same information as "fileName" for example.

2 ^ | v • Reply • Share ›



**Marcos Douglas Santos** → Alex • 5 months ago

Yep. I understood and you're right.

^ | v • Reply • Share ›



**Invisible Arrow** → Marcos Douglas Santos • 5 months ago

I believe this is a general problem in all dynamically typed languages. Sometimes even with descriptive names, it is difficult to get the context right.

You just sort of "happen to know" what type it is based on documentation and usage examples.

Recently

when I was working on few Gradle build scripts (in groovy of course),

I

had named a variable "xmlNode". Having worked with it, I knew the type

was "groovy.util.Node", but it wasn't so obvious for a co-worker.

1 ^ | v • Reply • Share ›



**Yegor Bugayenko** author → Invisible Arrow • 5 months ago

Yeah, weak typing is a very bad idea. Even an anti-pattern, worth a personal article :)

1 ^ | v • Reply • Share ›



**Lambda Pool** • 5 months ago

it demonstrates that Ruby syntax sux.

2 ^ | v • Reply • Share ›



**Invisible Arrow** • 5 months ago

I was waiting for this article :)

I realized the part where some prefixes and suffixes are redundant because the scope is clear, like the example that you've given.

But when it comes to dealing with some real world entities such as "Credit Card", I tend to use shorted names like "cc" or "card" now, compared to previous names such as "creditCard".

The former two are shorter and are still quite understandable in the given scope, but some people would argue that "creditCard" is still clearer.

Do you consider this a strict rule or are there some exceptions which are still acceptable? I do see that sometimes, there are exceptions made to some rules in all coding standards because of the situation.

Would you consider this as one of them?

PS: On a side note, this rule has helped me in general avoiding complex scope, which resulted in refactoring code by splitting and composing objects better.

Ex. A complex object had fields like "statusCode" and "statusMessage" that naturally belonged to a object of their own like "Status" having "code" and "message" fields.

^ | v · Reply · Share ›



**John Ohno** → Invisible Arrow · 4 months ago

No way, 'cc' as a variable name is clearly a boolean indicating whether or not there's a creative commons license applied, meaning royalty payments can be skipped! Furthermore, 'card' refers to the URI of the current page in the mobile version of the app!

^ | v · Reply · Share ›



**Yegor Bugayenko** author → Invisible Arrow · 5 months ago

My IDE (IntelliJ) is configured to mark all variables in red color if their names don't mach `[a-z]{3,12}`, that's why I avoid any (!) compound names. [We're planning](#) to do the same in Qulice. So, I don't make any exceptions, just to avoid red color in my code :) When you use this rule for some time, you will get used to it and any time you catch "creditCard" in the code your instincts will raise a red flag. So, I would suggest to stick to this rule fanatically.

2 ^ | v · Reply · Share ›




**Invisible Arrow** → Yegor Bugayenko · 5 months ago

Yes, the same here :) I had started using your "settings.jar" from one of your earlier posts and I too hate the warnings that IntelliJ gives, which is why I started renaming my variables. Most importantly, I started thinking more about the code and it's structure. There was plenty to learn from the


More from [yegor256.com](https://yegor256.com)



 **yegor256.com** recently published


**There Can Be Only One Primary Constructor**

33 Comments  Recommend 

 **yegor256.com** recently published

**Constructors Must Be Code-Free**

68 Comments  Recommend 

 **yegor256.com** recently published

**My Favorite Software Books - Yegor Bugayenko**

13 Comments  Recommend 