

Code Health: To Comment or Not to Comment?

Monday, July 17, 2017

This is another post in our Code Health series. A version of this post originally appeared in Google bathrooms worldwide as a Google Testing on the Toilet episode. You can download a printer-friendly version to display in your office.

By Dori Reuveni and Kevin Bourrillion

While reading code, often there is nothing more helpful than a well-placed comment. However, comments are not always good. Sometimes the need for a comment can be a sign that the code should be refactored.

Use a comment when it is infeasible to make your code self-explanatory. If you think you need a comment to explain what a piece of code does, first try one of the following:

Introduce an explaining variable.

```
// Subtract discount from price.
finalPrice = (numItems * itemPrice)
    - min(5, numItems) * itemPrice * 0.1;
    min(5, numItems) * itemPric
    finalPrice = price - discount;
```

Extract a method.

```
// Filter offensive words.
for (String word : words) { ... }
filterOffensiveWords(words);
```

Use a more descriptive identifier name.

```
int width = ...; // Width in pixels. int widthInPixels = ...;
```

Add a check in case your code has assumptions.

```
// Safe since height is always > 0.
return width / height; checkArgument(height > 0);
return width / height;
```

There are cases where a comment can be helpful:

Reveal your intent: explain why the code does something (as opposed to what it does).

```
// Compute once because it's expensive.
```

Protect a well-meaning future editor from mistakenly "fixing" your code.

```
// Create a new Foo instance because Foo is not thread-safe.
```

Clarification: a question that came up during code review or that readers of the code might have.

```
// Note that order matters because...
```

Explain your rationale for what looks like a bad software engineering practice.

```
@SuppressWarnings("unchecked") // The cast is safe because...
```

On the other hand, **avoid comments that just repeat what the code does**. These are just noise:

```
// Get all users.
userService.getAllUsers();
// Check if the name is empty.
if (name.isEmpty()) { ... }
```





Labels: Code Health , Dori Reuveni , Kevin Bourrillion , TotT

10 comments:



Jay July 17, 2017 at 3:26:00 PM PDT

Same as the `widthInPixels` example really, but my pet peeve is always around units of time. Don't comment that `timeout` variable, just rename it

`timeoutMillis`, `timeoutSecs`, or whatever it is.

Reply

Replies



Unknown July 19, 2017 at 8:01:00 AM PDT

And using the correct type. Why using an int for timeout if you could use a TimeSpan? It exposes the unit of time directly at the moment you use it.

Reply



renoX July 19, 2017 at 6:44:00 AM PDT

If possible in your language, I'd say that you should also replace name by types:

using the name widthInPixels won't help you if you do widthInPixels = widthInMilimeter, but a strong type would prevent this.

Reply

Replies



Дмитрий July 26, 2017 at 11:14:00 AM PDT

A good point, but I'd like to add that the bigger the visibility scope of a variable, the more relevant this advice is. In method scope you will be perfectly fine with a primitive type and a descriptive name. In a bigger scope you might consider hiding the type as the implementation detail.

Reply



Unknown July 19, 2017 at 6:46:00 AM PDT

Amen, Jay. Timeouts should be described in their units and this should extend all the way to the user interface/command line.

All real-world physical quantities should be described or typed with their units.

Reply

Replies



Unknown July 24, 2017 at 8:46:00 AM PDT

Even better if you are using something like F#, which has the concept of units built into the language, so you can't mistakenly add feet and centimeters together without converting first.

Reply



Mykaelos July 25, 2017 at 8:19:00 AM PDT

Great advice! It's all about explaining the why, because code can often hide that.

Thanks for the excellent blog post.

Also, could you add the Code Health label to this? So it shows up with the other Code Health posts? Seriously loving this series.

Reply

testi Google Testing Bloggers July 25, 2017 at 12:59:00 PM PDT

Done! Thanks for the heads up!

Reply

Success October 12, 2017 at 7:14:00 AM PDT

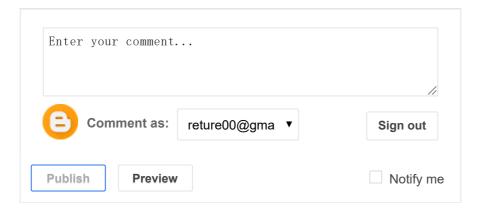
i think the timeoutMillis`will be more perfect.

Reply



This is really helpful and awesome practice to comment only when it is really required.

Reply



The comments you read and contribute here belong only to the person who posted them. We reserve the right to remove off-topic comments.









Google · Privacy · Terms