null==instance instead of instance==null [duplicate]

Asked 10 years, 9 months ago   Modified 10 years, 9 months ago

Viewed 521 times

3

This question already has answers here:

[what is the difference between null != object and object!=null](#) [duplicate] (2 answers)

Closed 10 years ago.

I have seen at many 3rd party code fragments where in some condition `null==instance` in used instead of `instance==null` like `if(null== connection)`.

Just curious, does this approach makes any impact on conditional statements or people are just cool to use it?

`java`

Share

Improve this question

Follow

edited Mar 24, 2014 at 12:47

Maroun
95.8k ● 30 ● 193 ● 247

asked Mar 24, 2014 at 12:43

Sachin Verma
3,802 ● 10 ● 45 ● 75

3 [Yoda conditions](#) Using `if(constant == variable)` instead of `if(variable == constant)`, like `if(4 == foo)`. Because it's like saying "if blue is the sky" or "if tall is the man". [blog.codinghorror.com/new-programming-jargon](#) – [Soner Gönül](#) Mar 24, 2014 at 12:49 ✏

I'd expect the compiled bytecode to be exactly the same in both cases -- one `ifnull` instruction. – [Louis Wasserman](#) Mar 24, 2014 at 16:42

4 Answers

Sorted by: Highest score (default) ⇅

▲

7

▼

🔖

✔

🕘

The most common reasons I hear quoted for using this are:

1. It's clever and cool.

2. It helps protect against assignment vs. comparison errors, since you can't assign to `null`.

I vehemently argue against the former, since "clever" very easily becomes "difficult to maintain" in any codebase. The latter has validity, though I think decent test coverage can accomplish the same task with more added value.

Personally I don't care for this style because it doesn't read correctly for me. Generally I like code to "read like prose" to make it easy to follow. And consider two prose statements:
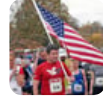
1. The object is empty.

2. Empty is the object.

The former sounds more natural to me.

Share   Improve this answer

Follow

David
218k ● 39 ● 215 ● 310

+1 for the second reason. – Maroun Mar 24, 2014 at 12:46

@David Thanks for the answer and adding `vehemently` to my dictionary! god bless you! – Sachin Verma Mar 25, 2014 at 6:59

No, there is no any difference.

6

This is just a *code style*, that may appeal in some cases useful to some, when `null` is immediately manifested like a first value in conditional statement so makes it more explicit.

Also imagine in case when "instance" may appear not just an instance of type, but some expression, say `(()=> { .... })`. You need in that case, first read an expression to the end, after see what is a condition, instead you see condition in first place.

Share   Improve this answer

Follow

Maroun
95.8k ● 30 ● 193 ● 247

▲

5

▼

🔖

🕘

I think it comes from C/C++ where you could accidentally assign a value without the compiler noticing:

```
if(variable = 0) { ... }
```

This causes no error and the condition is always false and the variable is assigned 0. So people started to strictly use the other form:

```
if(0 = variable) { ... }
```

which causes an error since 0 can not be assigned a value.

The notation `if(null == variable)` may be readable enough, but what about `if(b == a)` where **b is a constant** and **a a variable**? In this case, readability is a problem.

Share Improve this answer

Follow

edited Mar 24, 2014 at 13:02

answered Mar 24, 2014 at 12:48

But you would get this warning: `suggest parentheses around assignment used as truth value` ... and you are using warnings, right? – AKHolland Mar 24, 2014 at 15:16

@AKHolland IMHO using Yoda notation to avoid accidental assignment is obsolete with a modern compiler. As I said, this probably comes from a time where the compiler did **not** warn you. – helb Mar 24, 2014 at 15:21

No difference, except if you mistype and do `=` instead of `==` the compiler will complain as you can't assign to null.

4

Share Improve this answer

Follow

answered Mar 24, 2014 at 12:46

jpw
44.9k ● 6 ● 69 ● 90