Null object design pattern question

Ask Question

I recently watched this youtube tutorial on the Null Object design pattern. Even though there were some errors in it: such as the NullCar that doesn't do anything creates an infinite loop, the concept was well explained. My question is, what do you do when the objects that can be null have getters, and are used in your code? How do you know which value to return by default? Or should I implement this pattern inside all the objects? What if I need to return strings or primitives? I'm talking from a Java perspective.

EDIT: won't I be trading null objects testing for default value testing ? If not , why not ?

java design-patterns null-object-pattern

edited Jul 5 '12 at 7:41
abatishchev
66.7k 66 251 383

asked Nov 25 '08 at 19:56
Geo
42.9k 86 283 455

5 Answers

As far as I've understood it the idea is that the null object's value is as close to "nothing" as possible. That unfortunately means you have to define it yourself. As an example I personally use "" when I can't pass a null String, null object number for me is -1 (mostly because by default most database sequences start at 1 and we use those for item id:s a lot so -1 is dead giveaway it's a null object), with lists/maps/sets it's `Collections.EMPTY_SET`, `EMPTY_MAP` or `EMPTY_LIST` and so on and so forth. If I have custom class I have to create a null object from, I remove all actual data from it and see where that takes me and then apply what I just mentioned until it's "empty".

So you really don't "know" which value to return by default, you just have to decide it by yourself.

answered Nov 25 '08 at 20:10
P Arrayah
470 4 3

The objective of a Null Object is to avoid having Null references in the code. The values returned by Null Object getters depend on your domain. Zero or empty string are usually appropriate.

If we transpose the Null Object pattern to real life, what you're asking is similar to ask "*how old is nobody ?*".

Perhaps your design can be improved as you seem not to follow the tell, don't ask principle.

EDIT: the Null Object design pattern is typically used when an object delegates behavior to another object (such as in Strategy or State Design Patterns) ; as Tom Hawtin - tackline commented, use Special Case Objects for objects returning values.

edited Nov 26 '08 at 16:31

answered Nov 25 '08 at 20:14
philant
26.5k 10 56 99

I think it would be more accurate to comment that the context for the Null Object pattern is better met with "tell, don't ask" types. For value returning types, you are more likely to be heading for Special Case Objects. Possibly. – Tom Hawtin - tackline Nov 25 '08 at 20:27

2 Wow, a zen tao as programming hint... :-) – PhiLho Nov 26 '08 at 10:38

> what do you do when the objects that can be null have getters , and are used in your code ? How do you know which value to return by default ?

How do you know which classes to implement? This is a design question, it depends on the application.

Generally speaking the purpose of the NullObject pattern is to support a Replace Conditional with Polymorphism refactoring in the special case where the the conditional is a comparison against the null value of the programming language.

A correct implementation of the example in the video would require delegating the `driveCar` method to the `Car` classes. The `SlowCar` and `FastCar` classes would perform the loop, presumably through a shared implementation in a base class, and the `NullCar` would just return immediately.

In a Java context, the `NullCar.speed` attribute would probably be an unboxed int. So setting it to `null` is not an option. I would probably hide the attribute behind accessor, and have `NullCar.getSpeed` raise an exception. Any client code that would need a test to avoid this exception would instead move into the car classes.

Delegating all operations that directly depend on a speed value being available is an application of the Tell Don't Ask principle of object-oriented design mentioned by philippe

edited May 23 '17 at 12:24
Community ♦
1 1

answered Nov 25 '08 at 20:21
ddaa
37.1k 7 42 52

What should be the integration point of Null design pattern in code ? I think that DAO objects are the fisrt level client for this design pattern as they lookup an entity in database and return it simply.

The nullability check of these objects pollute the code in service layer or command layer where they are actually accessed and used.

Please comment.

answered Nov 28 '14 at 6:10
user1555669
321 3 3

It should return the null object for the class you are getting. For example, if you have a class `A` with a getter that returns an object of class `B` , then the corresponding `NullA` 's getter should return `NullB` .

answered Dec 16 '15 at 15:18
shawnhcorey
3,056 1 9 16