- Home
- Design Patterns
- Blog
- Products
- Sitemap
- Contact

AndyPatterns

Null Object Design Pattern

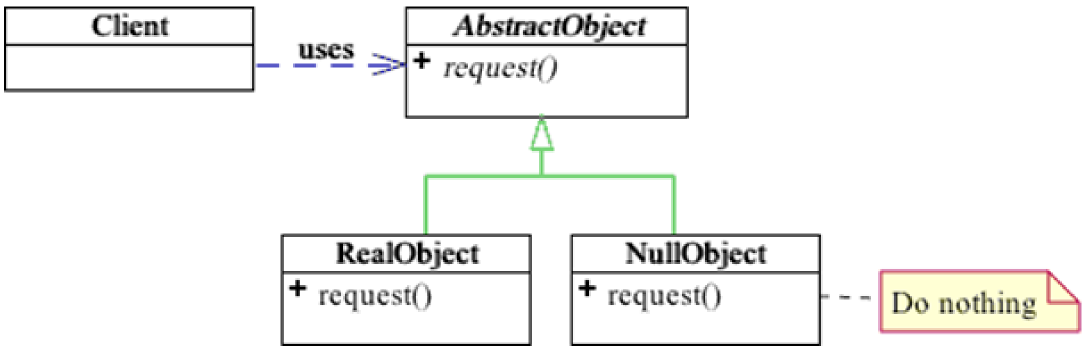
Sometimes I make the joke that design patterns are all about getting rid of if-else statements from your code. The null object pattern is an example of a pattern that does just that - check out the code at the bottom for details.

What is it?

A Null Object provides a surrogate for another object that shares the same interface, but does nothing.

This pattern was originally written up by Bobby Wolf, in Pattern Languages of Program Design 3.

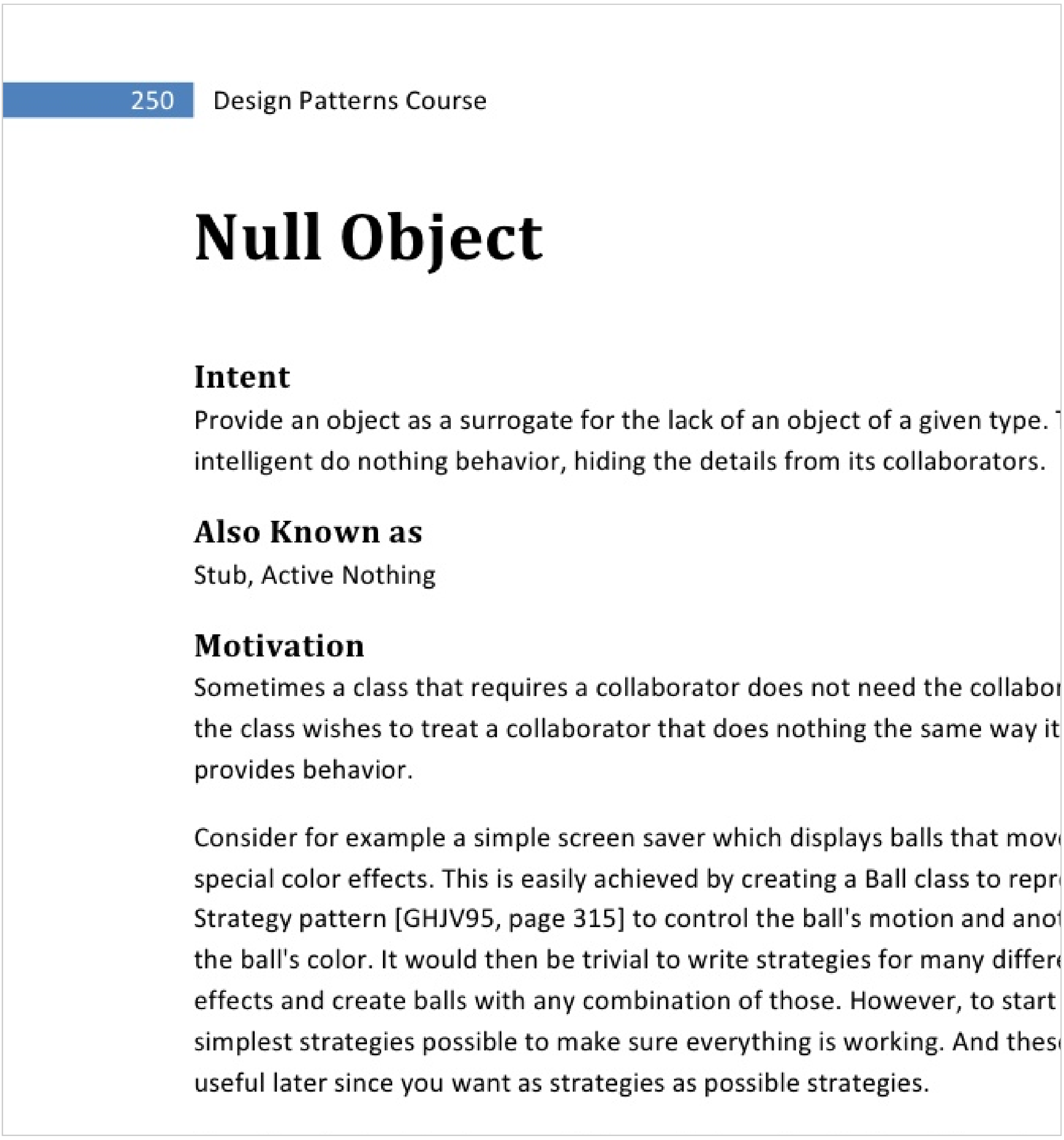
UML



Null Object Pattern

Have I used it?

Yes, I have used this pattern a few times in my work. You have to be a little bit careful about managing the swapping of null obect for the real thing. If bits of your code are pointing to the null object then you can't easily swap in the real thing. Better to put the null object / real object behind a proxy so that nobody knows what you are doing!



250 Design Patterns Course

Null Object

Intent

Provide an object as a surrogate for the lack of an object of a given type. T
intelligent do nothing behavior, hiding the details from its collaborators.

Also Known as

Stub, Active Nothing

Motivation

Sometimes a class that requires a collaborator does not need the collabor
the class wishes to treat a collaborator that does nothing the same way it
provides behavior.

Consider for example a simple screen saver which displays balls that mov
special color effects. This is easily achieved by creating a Ball class to repr
Strategy pattern [GHJV95, page 315] to control the ball's motion and anot
the ball's color. It would then be trivial to write strategies for many differe
effects and create balls with any combination of those. However, to start
simplest strategies possible to make sure everything is working. And these
useful later since you want as strategies as possible strategies.

[Null Object Design Pattern](#) from [tcab22](#)

Code:

Here is a python example. The so called API class is a real class with real functionality in it. We decide that we want to log calls to our API class (note the name of this class could be anything, and its got nothing to do with api's).

Without Null Object

```
from time import asctime, localtime

class AbstractObject: pass   # pretend python has abstract classes

class RealLogging:
    def Log(self, msg):
        print 'Logged at', asctime(localtime()), msg

# Proxy / wrapper around either null or real logger.

class Logger:
    def __init__(self):
        self.logger = RealLogging()
    def Log(self, msg):
        if self.logger:
            self.logger.Log(msg)
    def On(self):
        self.logger = RealLogging()
    def Off(self):
        self.logger = None
Logger = Logger()

# Usage:

class API:
    def doA(self):
        if Logger.logger:
            Logger.Log('Am calling A')
        print 'A done.'
    def doB(self):
        if Logger.logger:
            Logger.Log('Am calling B')
        print 'B done.'

o = API()
o.doA()
o.doB()

Logger.Off()
o.doA()
o.doB()
```

With Null Object

```
# Null Object Pattern

class AbstractLogging:
    def Log(self, msg): pass

from time import asctime, localtime

class RealLogging(AbstractObject):
    def Log(self, msg):
        print 'Logged at', asctime(localtime()), msg

class NullLogging(AbstractObject):
    def Log(self, msg):
        return

# Proxy / wrapper around either null or real logger.

class Logger:
    def __init__(self):
        self.On()
    def Log(self, msg):
        self.logger.Log(msg)
    def On(self):
        self.logger = RealLogging()
    def Off(self):
        self.logger = NullLogging()
Logger = Logger()

# Usage:

class API:
    def doA(self):
        Logger.Log('Am calling A')
        print 'A done.'
    def doB(self):
        Logger.Log('Am calling B')
        print 'B done.'

o = API()
o.doA()
o.doB()

Logger.Off()
o.doA()
o.doB()
```

Output

Without logging:

```
A done.
B done.
```

With logging:

```
Logged at Fri Jan 23 17:28:01 2009 Am calling A
A done.
Logged at Fri Jan 23 17:28:01 2009 Am calling B
B done.
```

Note:

Notice no more "if statements" in the client code (API class).