



[Home](#) / [Design Patterns](#) / [Behavioral patterns](#) / [Null Object](#)

# Null Object in Java



[Back to Null Object description](#)

Instead of using a null reference to convey absence of an object (for instance, a non-existent customer), one uses an object which implements the expected interface, but whose method body is empty. The advantage of this approach over a working default implementation is that a Null Object is very predictable and has no side effects: it does nothing.

For example, a function may retrieve a list of files in a folder and perform some action on each. In the case of an empty folder, one response may be to throw an exception or return a null reference rather than a list. Thus, the code which expects a list must verify that it in fact has one before continuing, which can complicate the design. By returning a null object (i.e. an empty list) instead, there is no need to verify that the return value is in fact a list. The calling function may simply iterate the list as normal, effectively doing nothing. It is, however, still possible to check whether the return value is a null object (e.g. an empty list) and react differently if desired.

The null object pattern can also be used to act as a stub for testing, if a certain feature such as a database is not available for testing.

```
class NullOutputStream extends OutputStream {
    public void write(int b) {
        // Do nothing
    }
}

class NullPrintStream extends PrintStream {
    public NullPrintStream() {
        super(new NullOutputStream());
    }
}

class Application {
    private PrintStream debugOut;
    public Application(PrintStream debugOut) {
        this.debugOut = debugOut;
    }

    public void doSomething() {
        int sum = 0;
        for (int i = 0; i < 10; i++) {
            sum += i;
            debugOut.println("i = " + i);
        }
        System.out.println("sum = " + sum);
    }
}

public class NullObjectDemo {
    public static void main(String[] args) {
        Application app = new Application(new NullPrintStream());
        app.doSomething();
    }
}
```

## Output

```
sum = 45
```

## Read next

This article is taken from our book **Design Patterns Explained Simply**.

All of the design patterns are compiled there. The book is written in clear, simple language that makes it easy to read and understand (just like this article).

We distribute it in PDF & EPUB formats so you can get it onto your iPad, Kindle, or other portable device immediately after your purchase.



 [Learn more](#)

### Code examples

Java	Null Object in Java	Null Object in Java

Design Patterns  
AntiPatterns  
Refactoring  
UML

My account  
Forum  
Contact us  
About us

© 2007-2018 SourceMaking.com  
All rights reserved.

[Terms / Privacy policy](#)