

# Null Object Pattern

- Provides an object as a surrogate for the lack of an object of a given type.
  - The Null Object Pattern provides intelligent do nothing behavior, hiding the details from its collaborators.
- NEW Forum Discussions: NEW  
 What Design Pattern Should I Choose?  
 Design Principles and Patterns Null Object Pattern  
 Motivation

There are some cases when a system has to use some functionality and some cases when it doesn't. Let's say we have to implement a class that should send the results to a log file or to the console. But this is just an additional option and the data is logged depending on the configuration values.

If there are cases when the client module does not have to log any data then it has to check the configuration parameter in and if block and then to call or not the Logger class. But as we know the 'if' block is not an elegant solution.

Intent

- Provide an object as a surrogate for the lack of an object of a given type.
- The Null Object Pattern provides intelligent do nothing behavior, hiding the details from its collaborators.

Implementation The participants classes in this pattern are:

- AbstractClass - defines abstract primitive operations that concrete implementations have to define.
- RealClass - a real implementation of the AbstractClass performing some real actions.
- NullClass - a implementation which do nothing of the abstract class, in order to provide a non-null object to the client.
- Client - the client gets an implementation of the abstract class and uses it. It doesn't really care if the implementation is a null object or an real object since both of them are used in the same way.

Applicability & Examples

Example: Log System Let's say we need a logging framework in order to support the logging of an application. The framework must fulfill the following requirements:

- The destination of the output messages should be selected from a configuration file and it can be one of the following options: Log File, Standard Console or Log Disabled.
- Must be open for extension; new logging mechanism can be added without touching the existing code.

Specific problems and implementation

Null Object and Factory

The Null Object design pattern is more likely to be used in conjunction with the Factory pattern. The reason for this is obvious: A Concrete Classes need to be instantiated and then to be served to the client. The client uses the concrete class. The concrete class can be a Real Object or a Null Object.

Null Object and Template Method

The Template method design pattern need to define an abstract class that define the template and each concrete class implements the steps for the template. If there are cases when sometimes template is called and sometimes not then, in order to avoid the checking a Null Object can be use to implement a Concrete Template that does nothing.

Removing old functionality

The Null Object can be used to remove old functionality by replacing it with null objects. The big advantage is that the existing code doesn't need to be touched.

Conclusion

The Null Object Pattern is used to avoid special if blocks for do nothing code, by putting the "do nothing" code in the Null Object which becomes responsible for doing nothing. The client is not aware anymore if the real object or the null object is called so the 'if' section is removed from client implementation.