Facade pattern

From Wikipedia, the free encyclopedia

The **facade pattern** (or **façade pattern**) is a software design pattern commonly used with object-oriented programming. The name is by analogy to an architectural facade.

A facade is an object that provides a simplified interface to a larger body of code, such as a class library. A facade can:

- make a software library easier to use, understand and test, since the facade has convenient methods for common tasks;
- make the library more readable, for the same reason;
- reduce dependencies of outside code on the inner workings of a library, since most code uses the facade, thus allowing more flexibility in developing the system;
- wrap a poorly designed collection of APIs with a single well-designed API (as per task needs).

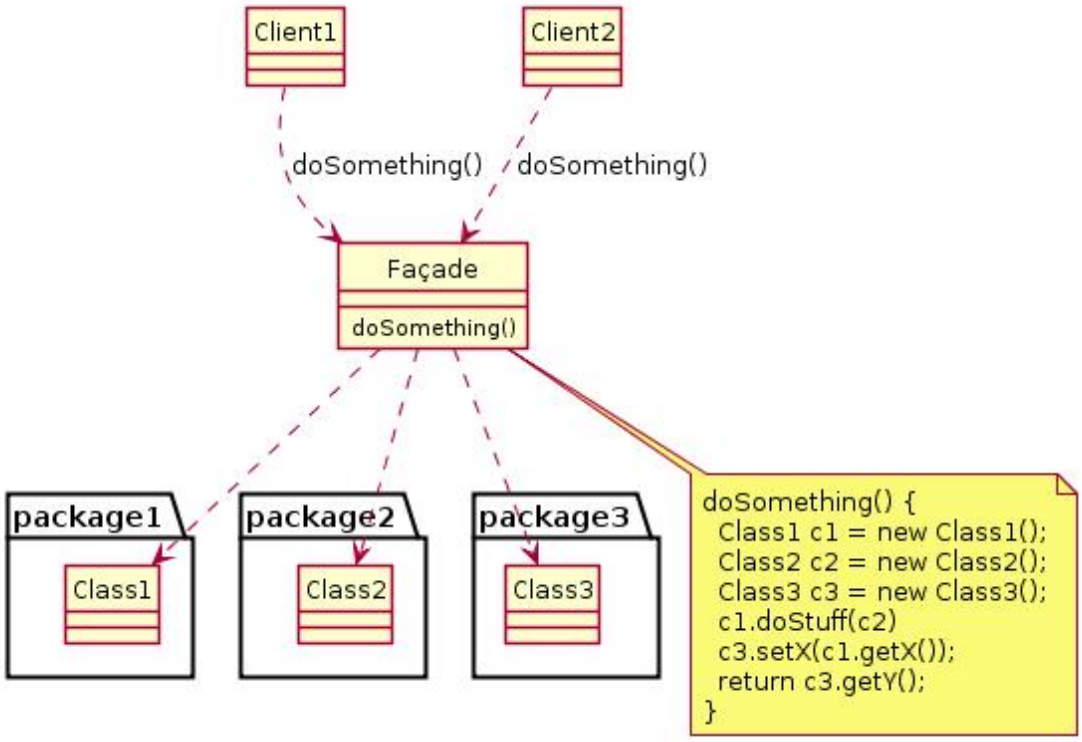
Contents

Usage

A Facade is used when one wants an easier or simpler interface to an underlying implementation object.[1] Alternatively, an adapter is used when the wrapper must respect a particular interface and must support polymorphic behavior. A decorator makes it possible to add or alter behavior of an interface at run-time.

Pattern	Intent
Adapter	Converts one interface to another so that it matches what the client is expecting
Decorator	Dynamically adds responsibility to the interface by wrapping the original code
Facade	Provides a simplified interface

Structure



Facade

The facade class abstracts Packages 1, 2, and 3 from the rest of the application.

Clients

The objects are using the Facade Pattern to access resources from the Packages.

Example

This is an abstract example of how a client ("you") interacts with a facade (the "computer") to a complex system (internal computer parts, like CPU and HardDrive).

```
/* Complex parts */

class CPU {
   public void freeze() { ... }
   public void jump(long position) { ... }
   public void execute() { ... }
}

class Memory {
   public void load(long position, byte[] data) { ... }
}

class HardDrive {
   public byte[] read(long lba, int size) { ... }
}

/* Facade */

class ComputerFacade {
   private CPU processor;
   private Memory ram;
   private HardDrive hd;
```

```
    public ComputerFacade() {
        this.processor = new CPU();
        this.ram = new Memory();
        this.hd = new HardDrive();
    }

    public void start() {
        processor.freeze();
        ram.load(BOOT_ADDRESS, hd.read(BOOT_SECTOR, SECTOR_SIZE));
        processor.jump(BOOT_ADDRESS);
        processor.execute();
    }
}

/* Client */

class You {
    public static void main(String[] args) {
        ComputerFacade computer = new ComputerFacade();
        computer.start();
    }
}
```

References

1. ^ Freeman, Eric; Freeman, Elisabeth; Kathy, Sierra; Bert, Bates (2004). Hendrickson, Mike; Loukides, Mike, eds. *Head First Design Patterns* (https://www.goodreads.com/book/show/58128.Head_First_Design_Patterns) (paperback) **1**. O'Reilly. pp. 243, 252, 258, 260. ISBN 978-0-596-00712-6. Retrieved 2012-07-02.

External links

- Description from the Portland Pattern Repository (http://c2.com/cgi/wiki?FacadePattern)

Retrieved from "http://en.wikipedia.org/w/index.php?title=Facade_pattern&oldid=606666802"
Categories: Software design patterns

The Wikibook *Computer Science Design Patterns* has a page on the topic of: ***Facade implementations in various languages***

Wikimedia Commons has media related to ***Facade pattern***.