**Facade Tasarım Kalıbı**

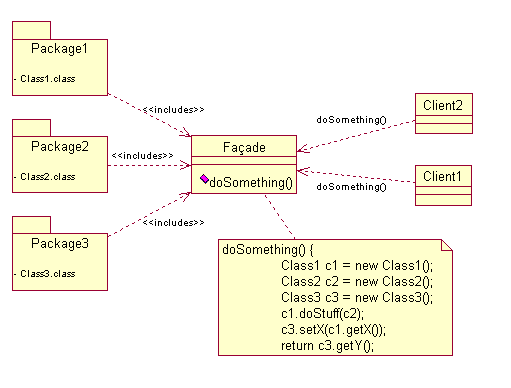
A facade is an object that provides a simplified interface to a larger body of code, such as a [class library](http://en.wikipedia.org/wiki/Class_library). A facade can:

* make a [software library](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/Coupling_(computer_programming)) 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](http://en.wikipedia.org/wiki/Application_programming_interface) with a single well-designed API (as per task needs).

## Usage

A Facade is used when one wants an easier or simpler interface to an underlying implementation object. Alternatively, an [adapter](http://en.wikipedia.org/wiki/Adapter_pattern) is used when the wrapper must respect a particular interface and must support [polymorphic](http://en.wikipedia.org/wiki/Polymorphism_(computer_science)) behavior. A [decorator](http://en.wikipedia.org/wiki/Decorator_pattern) makes it possible to add or alter behavior of an interface at run-time.

|  |  |
| --- | --- |
| **Pattern** | **Intent** |
| [Adapter](http://en.wikipedia.org/wiki/Adapter_pattern) | Converts one interface to another so that it matches what the client is expecting |
| [Decorator](http://en.wikipedia.org/wiki/Decorator_pattern) | Dynamically adds responsibility to the interface by wrapping the original code |
| Facade | Provides a simplified interface |



*\* 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();

}

}