**Facade**[**Design Pattern**](https://www.journaldev.com/1827/java-design-patterns-example-tutorial) is one of the **Structural design patterns** (such as [Adapter pattern](https://www.journaldev.com/1487/adapter-design-pattern-java)and [Decorator pattern](https://www.journaldev.com/1540/decorator-design-pattern-in-java-example)). Facade design pattern is used to help client applications to easily interact with the system

According to GoF Facade design pattern is:

Provide a unified interface to a set of interfaces in a subsystem. Facade Pattern defines a higher-level interface that makes the subsystem easier to use.

Suppose we have an application with set of interfaces to use MySql/Oracle database and to generate different types of reports, such as HTML report, PDF report etc.

But when the complexity increases or the interface behavior names are confusing, client application will find it difficult to manage it.

So we can apply Facade design pattern here and provide a [wrapper](https://www.journaldev.com/1002/wrapper-class-in-java) interface on top of the existing interface to help client application.

As you can see that using Facade pattern interface is a lot easier and cleaner way to avoid having a lot of logic at client side. JDBC Driver Manager class to get the database connection is a wonderful example of facade design pattern.

**Facade Design Pattern Important Points**

* Facade design pattern is more like a helper for client applications, it doesn’t hide subsystem interfaces from the client. Whether to use Facade or not is completely dependent on client code.
* Facade design pattern can be applied at any point of development, usually when the number of interfaces grow and system gets complex.
* Subsystem interfaces are not aware of Facade and they shouldn’t have any reference of the Facade interface.
* Facade design pattern should be applied for similar kind of interfaces, its purpose is to provide a single interface rather than multiple interfaces that does the similar kind of jobs.
* We can use [Factory pattern](https://www.journaldev.com/1392/factory-design-pattern-in-java) with Facade to provide better interface to client systems.

So we will have different set of interfaces to work with different types of database. Now a client application can use these interfaces to get the required database connection and generate reports.