**Façade Design Pattern**

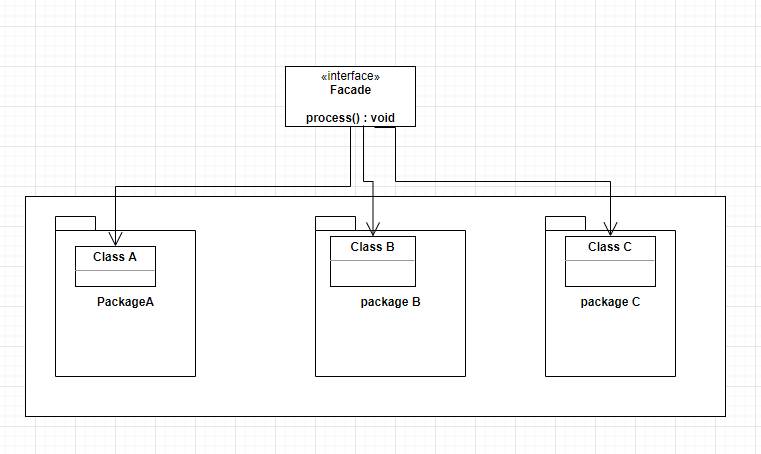
**Structural Design Pattern**

**Why Façade :**

Client has to interact with large number of interfaces and classes , so tightly coupled with those interfaces and classes. Façade Solves this problem.

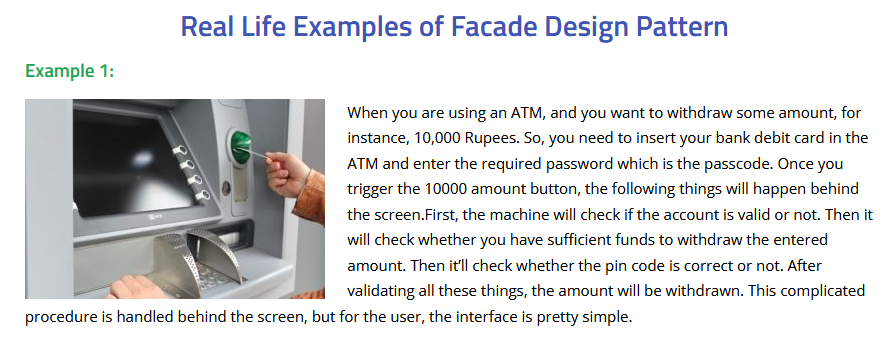
It hides the all complexities of subsystem.

In Façade , there is single interface which provide all functionality irrespective of client knows everything how it is implemented. Subsystem uses Packages.



**Example of Façade in Java :**

1. **Java.net.URL class has method openStream().**
2. In Java, the interface JDBC can be called a facade because, we as users or clients create connection using the “java.sql.Connection” interface, the implementation of which we are not concerned about. The implementation is left to the vendor of driver.

****

1. Abstract Factory can serve as an alternative to Façade when you only want to hide the way the subsystem objects are created from the client code.
2. Flyweight shows how to make lots of little objects, whereas Facade shows how to make a single object that represents an entire subsystem.
3. A Façade class can often be transformed into a Singleton since a single facade object is sufficient in most cases.

**Façade PartOne : Component Diagram :**

