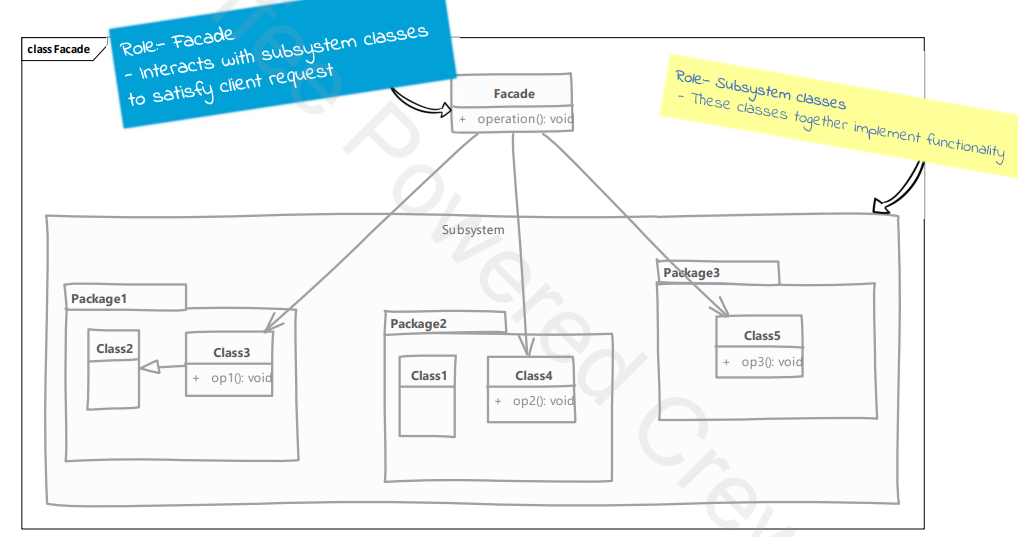
Facade Pattern

Structural Design Pattern

**What is Facade:**

1. Client has to interact with a large number of interfaces and classes in a subsystem to get results. So the client gets tightly coupled with those interfaces & classes.
2. This is a problem, because If you want to make changes to those classes or interfaces, you have to make sure that your client is going to work with those changes.
3. Facade solves this problem. Reduces the coupling of these classes and interfaces. Provides a simple and unified interface to a subsystem. Client interacts with just the facade not to get the same result. NOT one to one method forwarding to other classes.

**UML:**



**Implementation Considerations:**

1. Facade should minimize the complexity of the subsystem and provide a usable interface.
2. Facade is not a replacement for regular usage of classes in a subsystem. Those can be still used outside of the facade.
3. Great solution to simplify dependencies. Allows for weak coupling between subsystems.
4. If our only concern is coupling of client code to subsystem specific classes and not worried about simplification provided by the facade, then you can use an abstract factory pattern in place of the facade.

**Comparison with Adapter:**

1. Facade: intent is to simplify the usage of a subsystem for client code.
2. Facade: not restricted by any existing interface.
3. Adapter: is meant to simply adapt an object into a different interface.
4. Adapter: always written to conform to a particular interface expected by the client code.

**Examples:**

