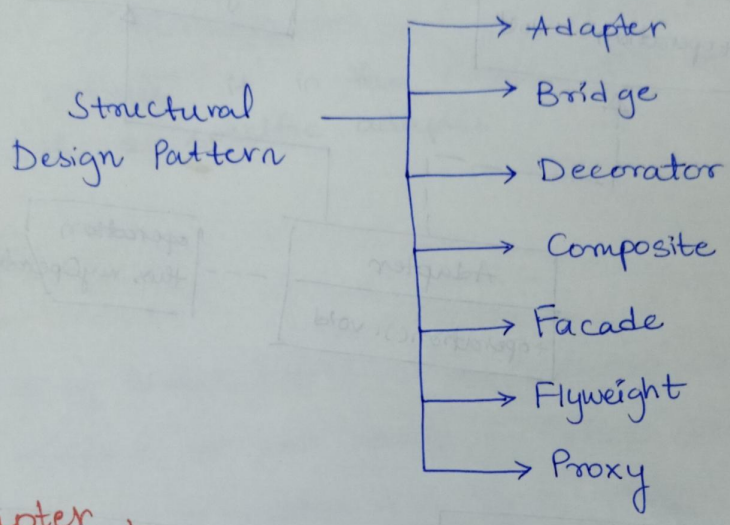
Structural Design Pattern :-

Structural dp deals with how classes and objects are arranged or composed.



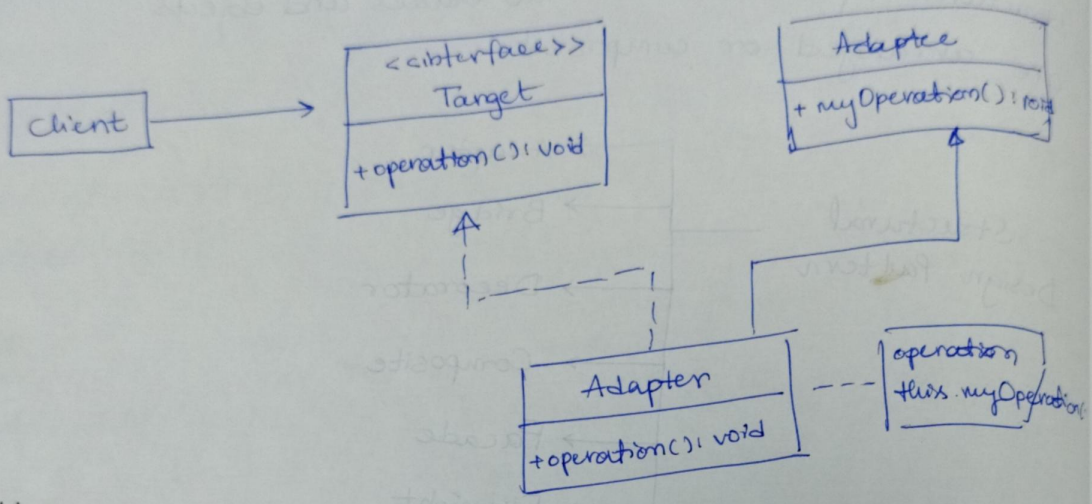
Adapter :-

What is Adapter?

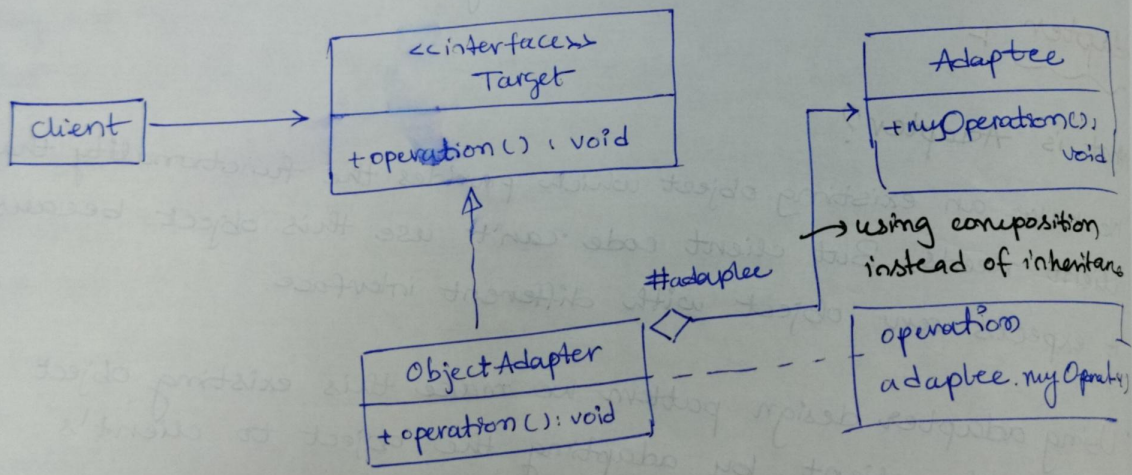
- We have an existing object which provides the functionality that client needs. But client code can't use this object because it expects an object with different interface.
- Using adapter design pattern we make this existing object work with client by adapting the object to client's expected interface.
- This pattern is also called as a wrapper as it "wraps" existing objects.

UML

- _____ (class adapter / Two way adapter).



UML _____ (object adapter)



Implement a Adapter

→ We start by creating a class for Adapter.

- Adapter must implement the interface expected by client.
- First we are going to toy out a class adapter by also extending from our existing class.
- In the class adapter implementation we're simply going to forward the method to another method inherited from adaptee.

- Next for object adapter, we are only going to implement target interface and accept adaptee as constructor argument in adapter i.e; make use of composition.

- An object adapter should take adaptee as an argument in constructor or as a less preferred soln; you can initialize it in the constructor thus tightly coupling with a specific adaptee.

Pitfalls :-

→ Using target interface and adaptee class to extend our adapter we can create a "class adapter" in java. However it creates an object which exposes unrelated methods in parts of your code, hence polluting it. Avoid class adapters! It is mentioned here only for sake of completeness.

→ It is tempting to do a lot of things in adapter besides simple interface translation. But this can result in an adapter showing different behaviour than the adapted object.

→ Not a lot of other pitfalls! As long as we keep them true to their purpose of simple interface translation they are good.