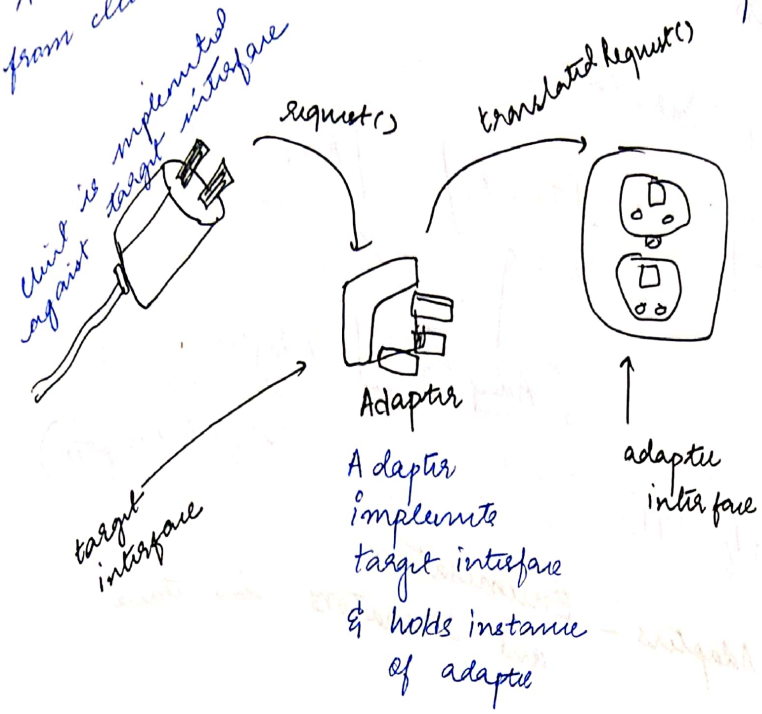
# Chapter 7: Adapter and Facade.

- Adapter acts as the middleman by receiving requests from client and converting them into requests.



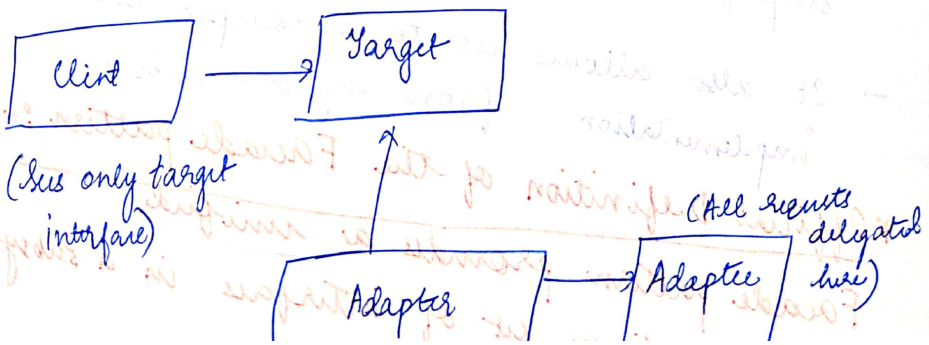

- Client and Adaptee are decoupled.
- No dumb questions

→ Job of implementing adapter & size of interface

→ Adapter's pattern is to convert one interface into another.

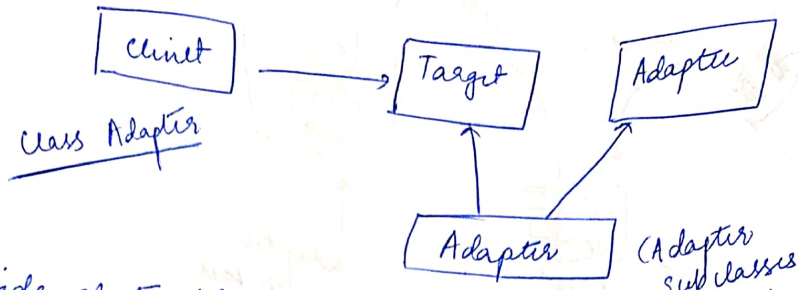
→ Two way Adapter ( so that adapter can act as old interface or new interface)

DP: The Adapter Pattern converts the interface of a class into another interface the clients expect. It lets classes work together that otherwise couldn't because of incompatible interfaces.



- Two types of adapters
  - Object
  - Class
    - Multiple Inheritance

*




Class Adapter

* Fireside chats. (Pg 341)

* Real world Adapters – Enumerators and Iterators in Java

Pg 350 (Fireside chats).

* Facade pattern : Simplifies the interface, as well as decouples a client from sub-system of components.

| Pattern | Intent * * |
|---|---|
| Decorator | connects one interface to another |
| Adapter | Doesn't alter interface, adds responsibility |
| Facade | Makes simpler |

→ This is the most important thing about a pattern.

- No dumb questions

→ Facades don't encapsulate, they provide a simplified interface. Still expose full funct.

→ It also allows us to decouple client implementation from any one subsystem.

* Official Definition of the Facade pattern :-

Facade pattern provides a unified interface to a set of interfaces in a subsystem. It defines a higher level interface.

* Principle of least knowledge : talk to only your immediate friends. (Law of Demeter)

* Brain power - 2 classes.

* Guidelins :
Invoke methods that belong to -
① Object itself
② Objects passed in as parameter to method
③ Any object method creats or instantiates
④ Any components of the object.

* Disadvantage of POIK (It results in more wrapper classes being written to handle method calls).

* System.out.println() violates

* Use Facade when :
① You want to provide simple interface to complex subsystem.
② Many dependencies between clients and implementation classes of abstraction.
③ Used to define entry point to each subsystem.

* Session Facade : It define a higher-level business component that contains & centralize complex interactions between lower level business components.

Adding a session facade decreases latency.