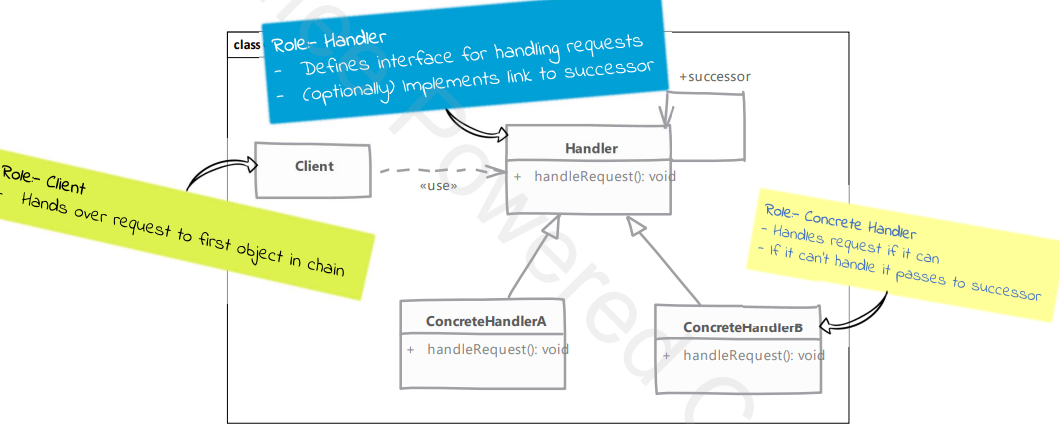
Chain of Responsibility

Behavioral Design Pattern

**What is Chain of Responsibility:**

1. When we need to avoid coupling the code which sends request to the code which handles that request.
2. Typically, the code which wants some request handled calls the exact method on an exact object to process it, thus the tight coupling. Chain of responsibility solves this problem by giving more than one object, chance to process the request.
3. Create objects which are chained together by one object knowing reference of object which is next in the chain. We give request to first object in chain, it can’t handle that it simply passes the request down the chain.

**UML:**



**Implementation steps:**

1. Defining handler interface/abstract class:
   1. Handler must define a method to accept incoming request
   2. Handler can define method to access successor in chain. It it’s an abstract class then we can even maintain successor.
2. Implement handler in one or more concrete handlers. Concrete handler should check if it can handle the request. If not, then it should pass request to next handler.
3. Create our chain of objects next. We can do it in the client.
4. Client needs to know only the first object in chain.

**Intent:** lets you pass requests along a chain of handlers. Upon receiving a request, each handler decides either to process the request or to pass it to the next handler in the chain.

**Structure:**

