**Chain of Responsibility Design Pattern**

Chain of responsibility design pattern is used to provide a chain of objects to handle the request.

**Intent**

* Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.

The intent is to chain the receiving objects to handle the request one by one till it gets handled.

**Problem**

* Sender object does not know which object is going to handle the request.

There may be a situation where the sender does not know which specific object has to handle its request.

**Solution**

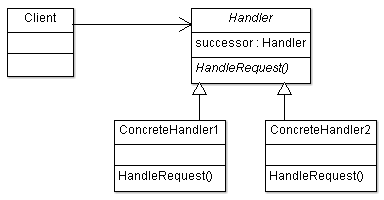
* Have multiple objects to handle the request of sender object.

So we can have multiple objects in chain and handle the request one by one till the request is handled.

**Where it is applicable?**

* There may be multiple object which can handle the request.
* The sender object does not know which object can handle its request.

**Structure**



**Participant classes**

* **Handler** class provides interface HandleRequest() to client. It also has member to keep the successor of handler object and method MakeSuccessor() to set the successor.
* The concrete classes **ConcreteHandler1** and **ConcreteHandler2** implement the method HandleRequest().

**How they work together?**

* Client requests to handle the request, ConcreteHandler handles the request if it can, otherwise it passes it to another object in chain through successor information. So all the objects in chain one by one try to handle the request in same way till last object.
* So we can see it decouples sender and requester. Also it provides flexibility to add further responsibility in handler. There may be cases when the request cannot be handled at all.