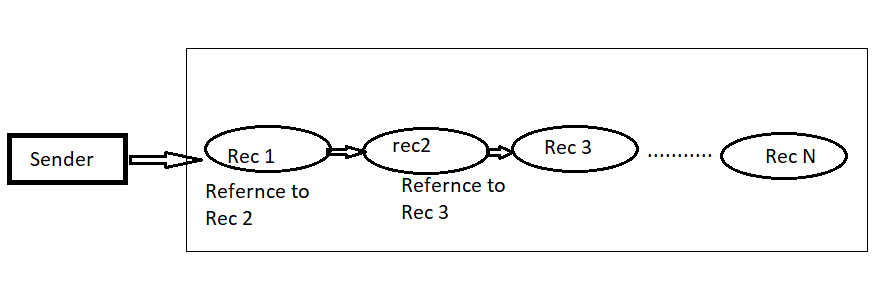
**Chain of responsibility –Design Pattern:**

🡪comes under Behavioral Pattern

🡪As the name suggests🡪 this pattern creates a chain of receivers objects for a request

🡪 this pattern decouples senders and receiver of request based on the type of request

🡪in this patter , normally each receiver contains the reference to another receiver

🡪If one of the obj cannot handle the request , then it passes the same to the next receiver and so on

🡪One receiver in the chain handles a request .. Eg Quiz master

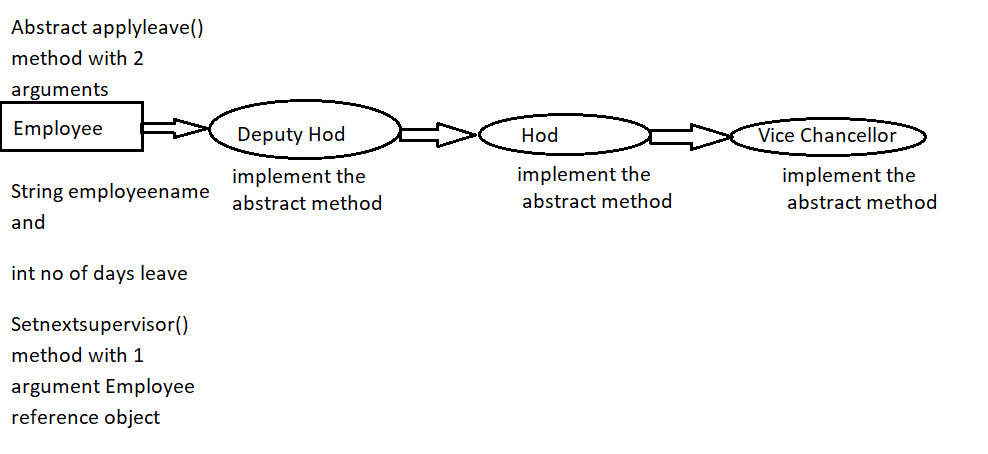
(or)

one or more receiver in the chain handles the request Eg. ATM Machine / Leave Application

**Implementation: Leave application**

Employee -- > Deputy HOD -- > HOD --> Vice Chancellor

Apply leave 🡪 Approve 10 days leave 🡪 Approve 20 days leave 🡪 Approve 30 days leave



1. Create an abstract class Employee with
   1. Abstract applyleave() method with 2 arguments

String employeename and

int no of days leave

* 1. Setnextsupervisor() method with 1 argument Employee reference object

1. The above abstract class is extended by
   1. Deputy Hod class
   2. Hod class and
   3. Vice chancellor class

All should implement the above abstract method

**Employee.java**

public abstract class Employee {

protected Employee supervisor;

public void setNextSupervisor(Employee supervisor) {

this.supervisor = supervisor;

}

public abstract void applyLeave(String empname, int no\_of\_days\_leave);

}

**DeputyHead.java**

public class DeputyHead extends Employee{

private final int Max\_leave\_can\_approve=10;

@Override

public void applyLeave(String empname, int no\_of\_days\_leave) {

if( no\_of\_days\_leave<=Max\_leave\_can\_approve)

{

System.out.println("Deputy Hod approved "+ no\_of\_days\_leave+" days leave for the employee.."+empname);

}

else

{

supervisor.applyLeave(empname, no\_of\_days\_leave);

}

}

}

**Head.java**

public class Head extends Employee{

private final int Max\_leave\_can\_approve=20;

@Override

public void applyLeave(String empname, int no\_of\_days\_leave) {

if( no\_of\_days\_leave<=Max\_leave\_can\_approve && no\_of\_days\_leave>10)

{

System.out.println("Hod approved "+ no\_of\_days\_leave+" days leave for the employee.."+empname);

}

else

{

supervisor.applyLeave(empname, no\_of\_days\_leave);

}

}

}

**Vice\_Chancellor.java**

public class Vice\_Chancellor extends Employee{

private final int Max\_leave\_can\_approve=30;

@Override

public void applyLeave(String empname, int no\_of\_days\_leave) {

if( no\_of\_days\_leave<=Max\_leave\_can\_approve)

{

System.out.println("VC approved "+ no\_of\_days\_leave+" days leave for the employee.."+empname);

}

else

{

System.out.println("Leave got cancelled.. Meet VC directly");

}

}

}

**ChainResponPaternDemo.java**

public class ChainResponPaternDemo {

public static void main(String[] args) {

Employee e;

DeputyHead dh =new DeputyHead();

e=dh;

Head h= new Head();

Vice\_Chancellor vc =new Vice\_Chancellor();

dh.setNextSupervisor(h);

h.setNextSupervisor(vc);

e.applyLeave("Suba", 10);

e.applyLeave("Subasatheesh", 20);

e.applyLeave("Eugin", 30);

e.applyLeave("John", 40);

}

}

**Output:**

Deputy Hod approved 10 days leave for the employee..Suba

Hod approved 20 days leave for the employee..Subasatheesh

VC approved 30 days leave for the employee..Eugin

Leave got cancelled.. Meet VC directly