

Behavior Design Pattern: Chain Of Responsibility



In the above scenario, we take loanAmount from the controller and decide which user will process that.

If loanAmount ≤ 200000 , Executive can process the loan

If loanAmount ≤ 500000 , AsstManager can process the loan

If loanAmount ≤ 1000000 , Manager can process the loan

Else: above 10 lakh, loan is not permitted

We first create a LoanHandler class that gives the method to define loan handlers for these classes.

```
public abstract class LoanHandler {
    LoanHandler loanHandler;

    public void forwardHandler(LoanHandler loanHandler){ //this allows controller to define handlers
        this.loanHandler = loanHandler;
    }

    public abstract void applyLoan(String acctNumber, double loanAmount);
}
```

This is how controller defines handlers

```
//rules of forward handling
executive.forwardHandler(assistantManager); //AsstManager is Handler of executive
assistantManager.forwardHandler(manager); //Manager is handler of AsstManager
```

In any class, check for the amount and then apply loan if amount is more than permissible.

```
public class Executive extends LoanHandler{ // AssistantManager is executives loanHandler
    @Override
    public void applyLoan(String acctNumber, double loanAmount) {
        if(loanAmount <= 200000) {
            System.out.println("Loan processed by executive ");
        }
        else
            loanHandler.applyLoan(acctNumber, loanAmount); //assiMan.loanHandler()
    }
}
```