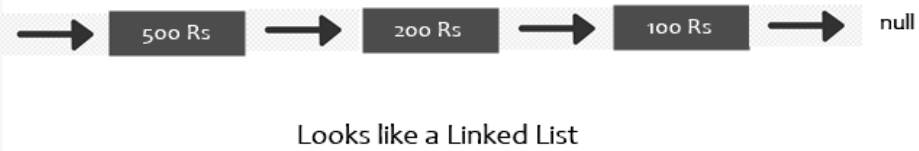
Chain of Responsibility

'The ATM Dispenser'



Looks like a Linked List

The Request object – It contains the actual Amount to be dispensed.

```
public class Amount {

    private int amount;

    public Amount(int amount) { this.amount=amount; }

    public int getAmount() { return this.amount; }
}
```

Interface/Contract for all the Dispenser objects – 500, 200 and 100.

```
public interface Dispenser {

    void setNext(Dispenser next);

    void dispense(Amount amount);
}
```

Dispenser Implementations – 500, 200 and 100

```
public class Rupee500Dispenser implements Dispenser {

    private Dispenser chain;

    @Override
    public void setNext(Dispenser next) { this.chain= next; }

    @Override
    public void dispense(Amount amount) {
        if(amount.getAmount() >= 500){
            int numNotes = amount.getAmount()/500;
            int remainingAmt = amount.getAmount() % 500;
            System.out.println("Dispensing ["+numNotes+"] - 500 rupee note(s).");

            if(remainingAmt !=0) this.chain.dispense(new Amount(remainingAmt));
        }else{
            // If amount is less than 500 as you might have requested
            // say, 400 Rs to be dispensed, you can't get 500 rupee note, so move
            // the responsibility to the next object.
            this.chain.dispense(amount);
        }
    }

}
```

```
public class Rupee200Dispenser implements Dispenser {

    private Dispenser chain;

    @Override
    public void setNext(Dispenser next) { this.chain= next; }

    @Override
    public void dispense(Amount amount) {
        if(amount.getAmount() >= 200){
            int numNotes = amount.getAmount()/200;
            int remainingAmt = amount.getAmount() % 200;
            System.out.println("Dispensing ["+numNotes+"] - 200 rupee note(s).");

            if(remainingAmt !=0) this.chain.dispense(new Amount(remainingAmt));
        }else{
            // If amount is less than 200 as you might have requested
            // say, 100 Rs to be dispensed, you can't get 200 rupee note, so move
            // the responsibility to the next object.
            this.chain.dispense(amount);
        }
    }

}
```

```
public class Rupee100Dispenser implements Dispenser {

    private Dispenser chain;

    @Override
    public void setNext(Dispenser next) { this.chain= next; }

    @Override
    public void dispense(Amount amount) {
            int numNotes = amount.getAmount()/100;
            System.out.println("Dispensing ["+numNotes+"] - 100 rupee note(s).");
    }

}
```

Reciever Class – The request to dispense the amount would come to this class. You can call it as the starting point of the chain logic.

```
public class ATM {

    private Dispenser dispenser500;

    public ATM() {
        this.dispenser500 = new Rupee500Dispenser();
        Dispenser dispenser200 = new Rupee200Dispenser();
        Dispenser dispenser100 = new Rupee100Dispenser();

        dispenser500.setNext(dispenser200);
        dispenser200.setNext(dispenser100);
    }

    public void dispense(Amount amount) { dispenser500.dispense(amount); }


}
```

It's now time to test our code 😊

```
import java.util.Scanner;

public class TestATMDispenser {
       public static void main(String[] args) {
           ATM atm = new ATM();

           while (true) {
               int amount;
               System.out.println("Please enter the amount!!!");
               Scanner input = new Scanner(System.in);
               amount = input.nextInt();

        // As you can just get 500,200 or 100 rupee notes, amount like 1250 Rs or 50 Rs or 1999 Rs can't be processed
               if (amount % 100 != 0) {
                   System.out.println("Unable to dispense. Please put the amount in multiple of 100s.");
                   continue;
               }

               // This is the starting point of actual processing
               atm.dispense(new Amount(amount));
           }

       }
}
```

Test Output

```
Please enter the amount!!!
1111
Unable to dispense. Please put the amount in multiple of 100s.
Please enter the amount!!!
1100
Dispensing [2] - 500 rupee note(s).
Dispensing [1] - 100 rupee note(s).
Please enter the amount!!!
2000
Dispensing [4] - 500 rupee note(s).
Please enter the amount!!!
2300
Dispensing [4] - 500 rupee note(s).
Dispensing [1] - 200 rupee note(s).
Dispensing [1] - 100 rupee note(s).
Please enter the amount!!!
500
Dispensing [1] - 500 rupee note(s).
Please enter the amount!!!
10000
Dispensing [20] - 500 rupee note(s).
```