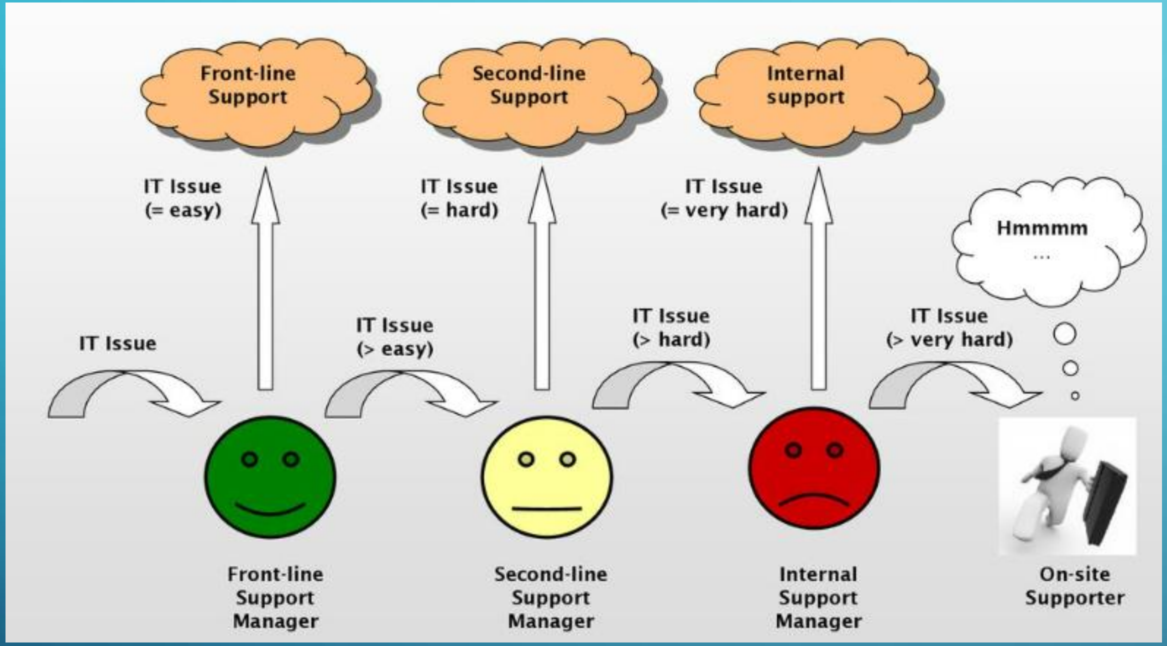
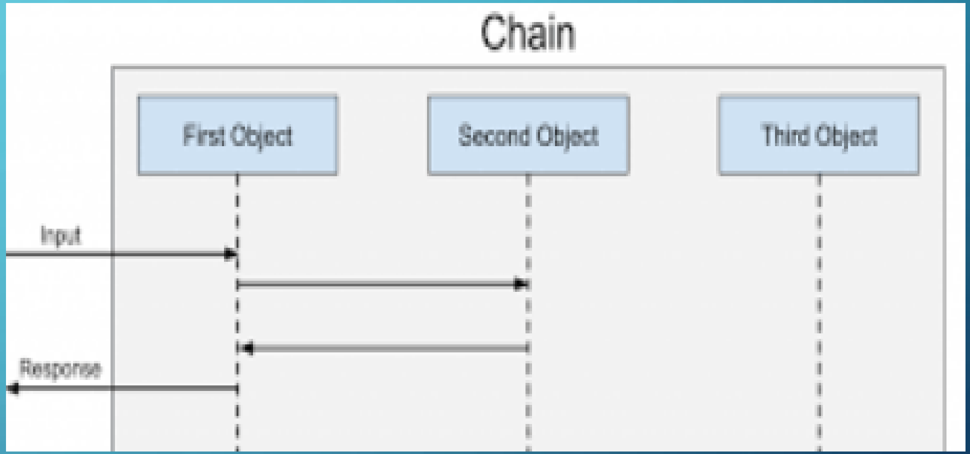
CHAIN OF RESPONSIBILITY PATTERN

AMIR DIRIN

SPRING 2023

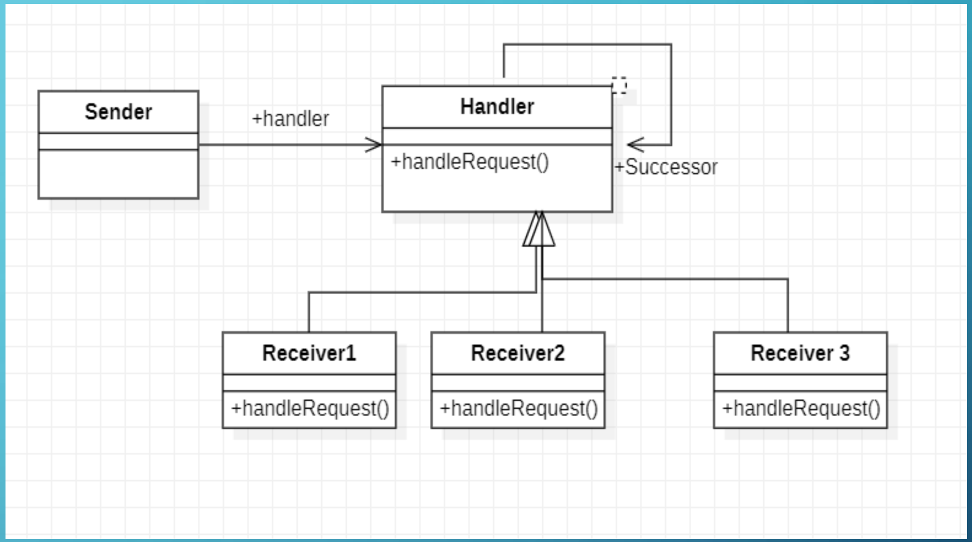
CHAIN

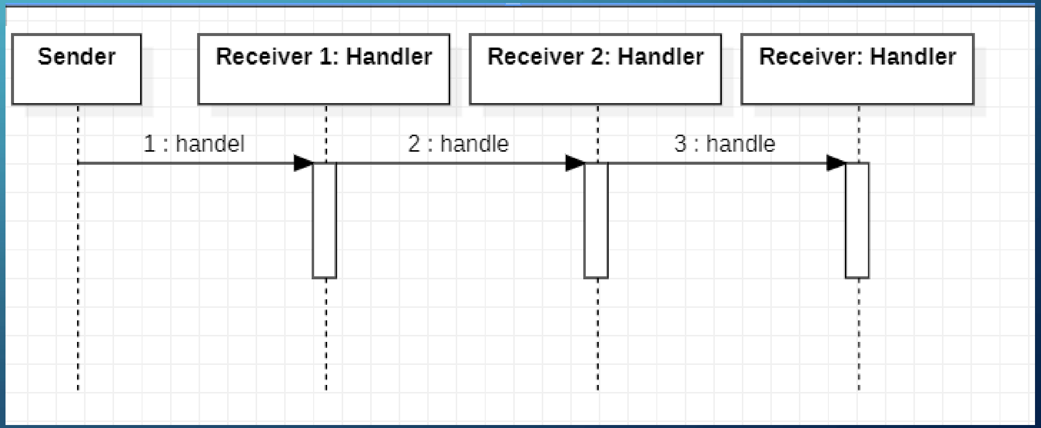

Front-line Support
Second-line Support
Internal support
IT Issue (= easy)
IT Issue (= hard)
IT Issue (= very hard)
Hmmmm ...
IT Issue
IT Issue (> easy)
IT Issue (> hard)
IT Issue (> very hard)
Front-line Support Manager
Second-line Support Manager
Internal Support Manager
On-site Supporter


Chain
First Object
Second Object
Third Object
Input
Response

CHAIN OF RESPONSIBILITY

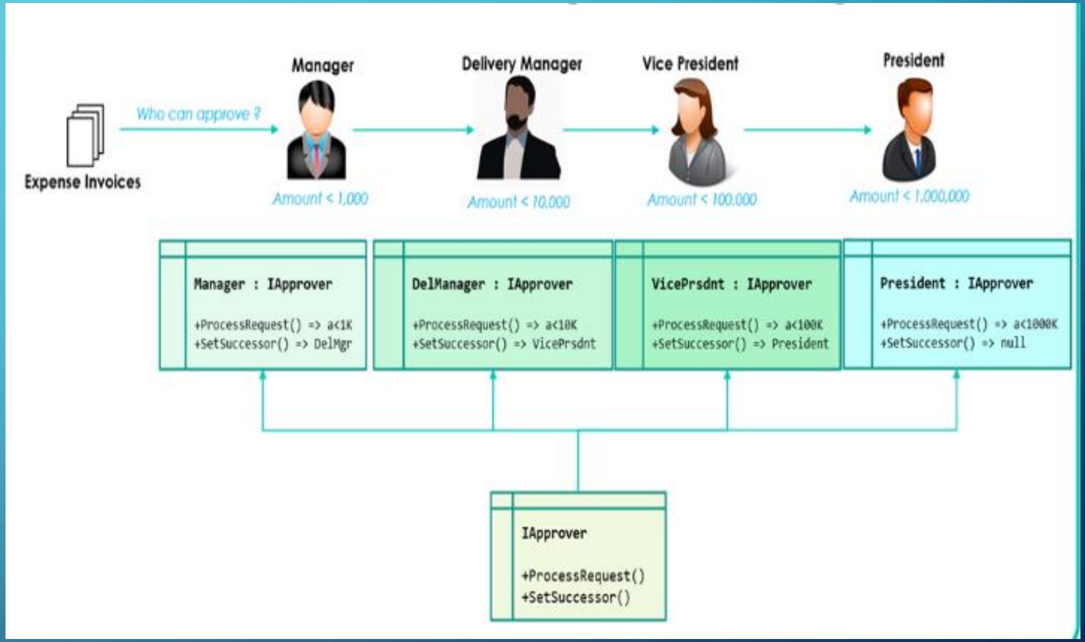
- In chain of responsibility, sender sends a request to a chain of objects.
- The request can be handled by an object in the chain.
  - If one object cannot handle the request then it passes the same to the next receiver and so on.
  - We send a request to a chain of receivers without having to know which one handle the request.



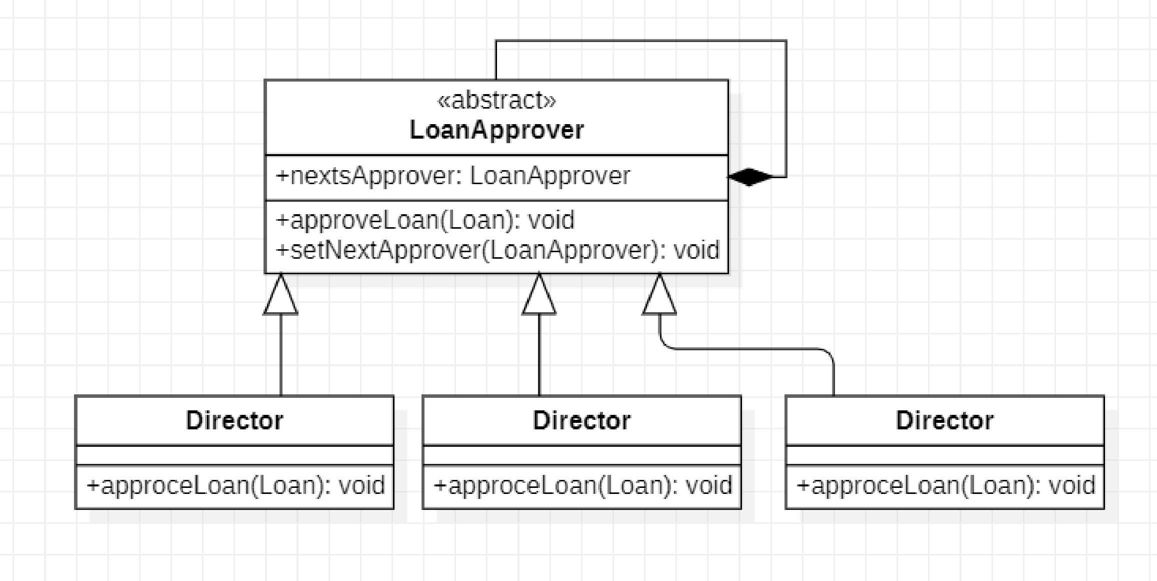


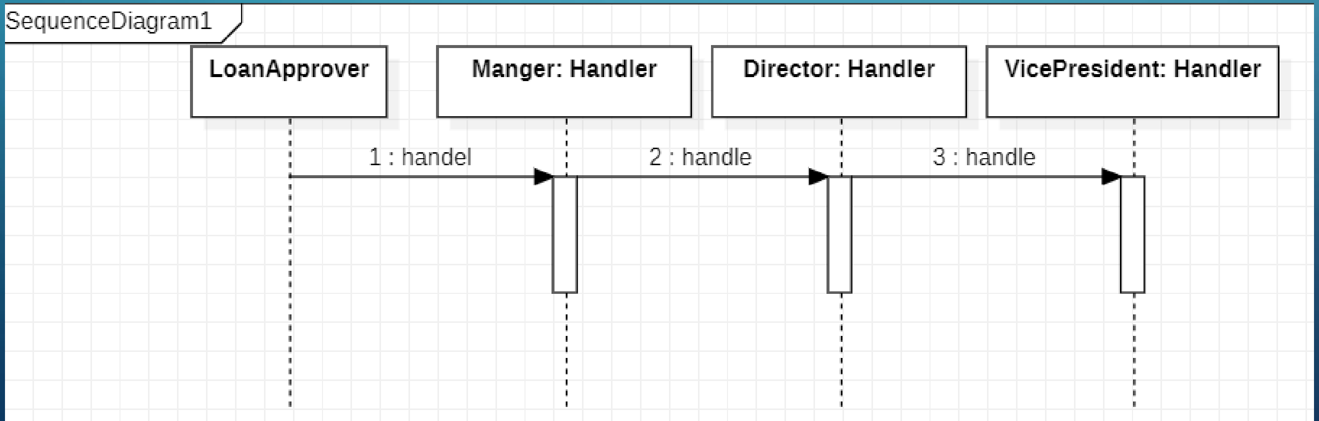
LOAN APPROVAL (LECTURE DEMO)

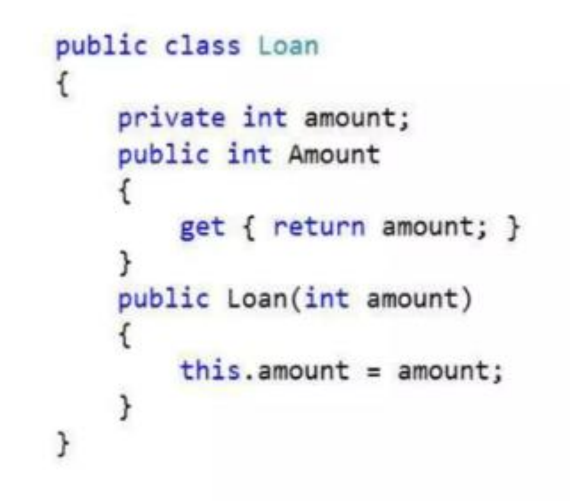
- In a bank where the approval route for mortgage applications are from the bank manager to the director then to the vice president, where the approval limits are
  - Manger – 0 to 100k
  - Director – 100k to 250k
  - Vice president- anything above 250k
- We will pass the request to the manager until the application is processed.
- Add use case, class and sequence diagrams.



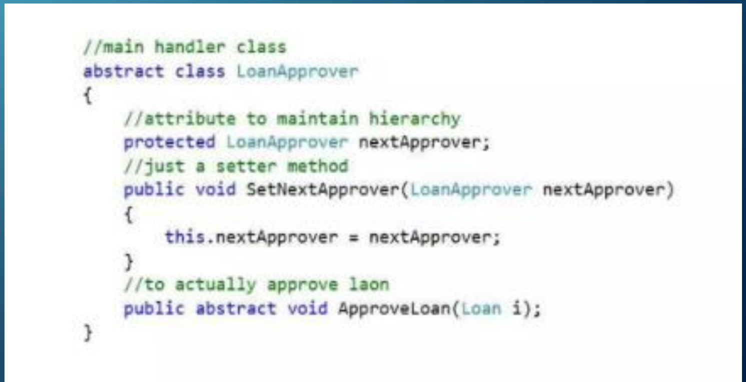
SOLUTION





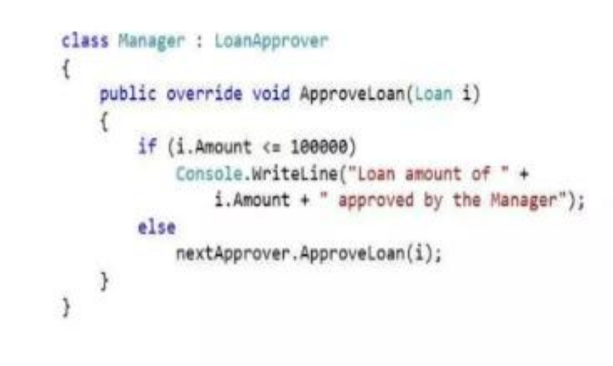


```
public class Loan
{
    private int amount;
    public int Amount
    {
        get { return amount; }
    }
    public Loan(int amount)
    {
        this.amount = amount;
    }
}
```

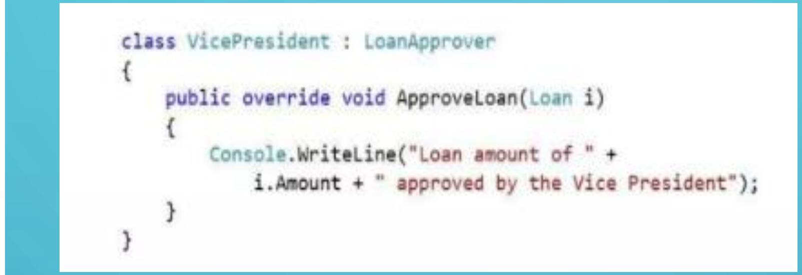


```
//main handler class
abstract class LoanApprover
{
    //attribute to maintain hierarchy
    protected LoanApprover nextApprover;
    //just a setter method
    public void SetNextApprover(LoanApprover nextApprover)
    {
        this.nextApprover = nextApprover;
    }
    //to actually approve laon
    public abstract void ApproveLoan(Loan i);
}
```

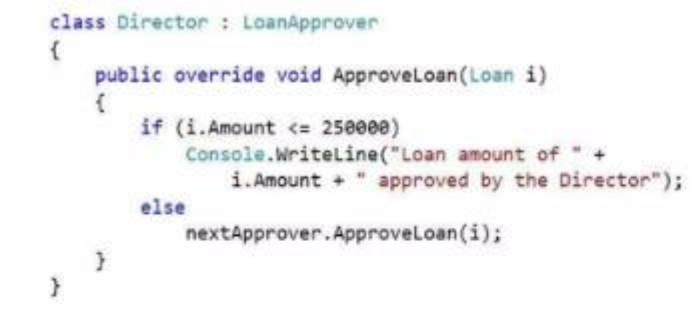
CLASSES



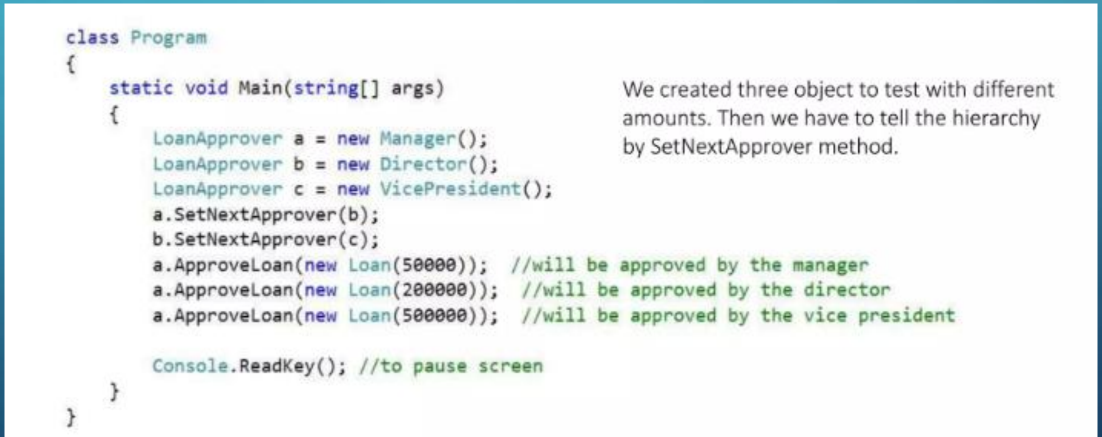
```
class Manager : LoanApprover
{
    public override void ApproveLoan(Loan i)
    {
        if (i.Amount <= 100000)
            Console.WriteLine("Loan amount of " +
                i.Amount + " approved by the Manager");
        else
            nextApprover.ApproveLoan(i);
    }
}
```



```
class VicePresident : LoanApprover
{
    public override void ApproveLoan(Loan i)
    {
        Console.WriteLine("Loan amount of " +
            i.Amount + " approved by the Vice President");
    }
}
```



```
class Director : LoanApprover
{
    public override void ApproveLoan(Loan i)
    {
        if (i.Amount <= 250000)
            Console.WriteLine("Loan amount of " +
                i.Amount + " approved by the Director");
        else
            nextApprover.ApproveLoan(i);
    }
}
```



```
class Program
{
    static void Main(string[] args)
    {
        LoanApprover a = new Manager();
        LoanApprover b = new Director();
        LoanApprover c = new VicePresident();
        a.SetNextApprover(b);
        b.SetNextApprover(c);
        a.ApproveLoan(new Loan(50000));  //will be approved by the manager
        a.ApproveLoan(new Loan(200000));  //will be approved by the director
        a.ApproveLoan(new Loan(500000));  //will be approved by the vice president

        Console.ReadKey(); //to pause screen
    }
}
```

We created three object to test with different amounts. Then we have to tell the hierarchy by SetNextApprover method.

LECTURE ASSIGNMENT

- ATM System to windthrow 20, 50, 100 Euros
- Apply for leave 3 days, a week and more then a week
- Hotel Room
- Security Monitoring System