OOAD LAB 9&10

NAME : SAHANA RAO	SRN: PES1UG20CS588	SECTION: J

1. **Problem statement**

Given a company's Leave system management which has 3 kinds of leaves: CL, SL and VL each having its own conditions. The leaves are approved by a hierarchy of management staff based on conditions. We are required to represent the design (using appropriate design patterns) in a UML Class Diagram and implement the same.

2. **Design patterns considered**

The design patterns considered in our approach are mainly **creational** and **behavioural**. Under creational we are looking forward to use Factory pattern and in behavioural we are using chain of responsibility.

What is Creational pattern and factory pattern?

Creational design patterns deals with object creation mechanisms, trying to create objects in a manner suitable to the situation.

Factory pattern is a creational design pattern that provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created. It is a way to encapsulate the creation of objects.

What is behavioural pattern and chain of responsibility?

Behavioural patterns help to identify common communication patterns among objects and define how those objects should interact with one another.

Chain of Responsibility is one of the behavioural design patterns. It allows you to pass requests along a chain of handlers until one of them can handle the request.

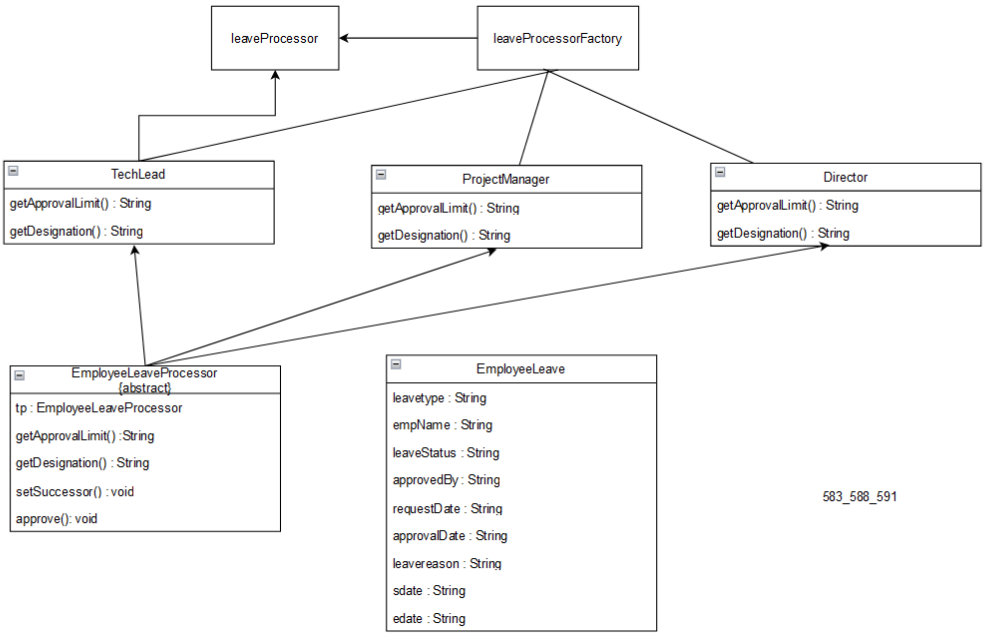
3. **Design patterns used**

Factory pattern: The LeaveProcessorFactory class creates objects of the EmployeeLeaveProcessor type based on the input argument type. This factory pattern allows the client code to create objects without knowing the specific

class of the object that will be created, thereby providing an abstraction layer between the client code and the object creation process.

Chain of Responsibility: The design pattern used in this code is the Chain of Responsibility pattern. The EmployeeLeaveProcessor abstract class and its concrete subclasses TechLead, ProjectManager, and Director implement the chain of responsibility pattern, where each object in the chain has a reference to the next object in the chain. The Client (leave_man class) sends a request to the first object in the chain (TechLead), and the object decides whether it can handle the request or not. If it can handle the request, it processes it; otherwise, it passes the request to the next object in the chain (ProjectManager), and the process continues until the request is handled or there is no more object in the chain.

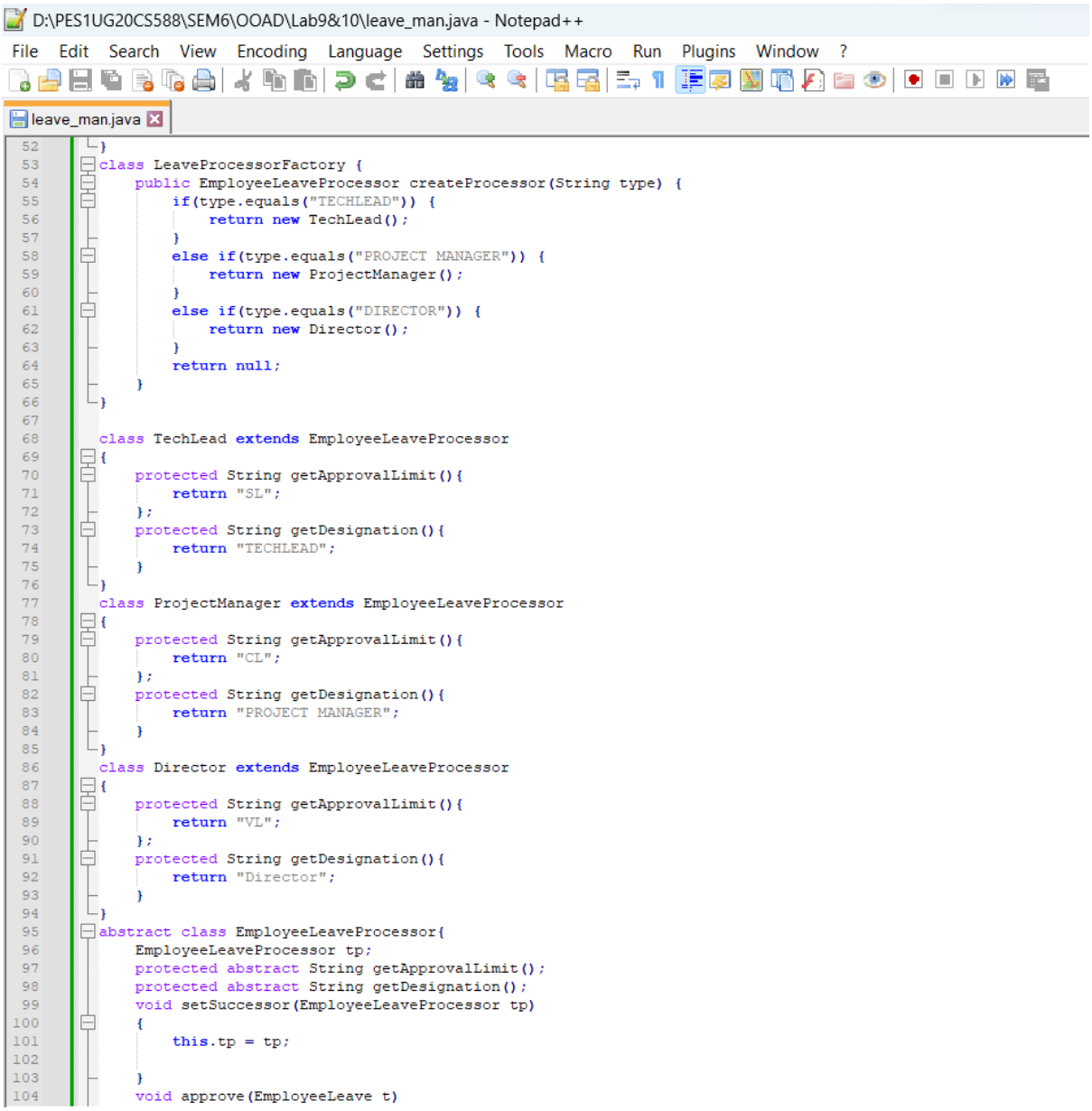
4. UML Class Model



5. Code

leave_man.java

```
import java.util.*;

class leave_man{
    public static void main(String[] args)
    {
        LeaveProcessorFactory factory = new LeaveProcessorFactory();
        TechLead tlead = (TechLead)factory.createProcessor("TECHLEAD");
        ProjectManager pm = (ProjectManager)factory.createProcessor("PROJECT MANAGER");
        Director dir = (Director)factory.createProcessor("DIRECTOR");

        tlead.setSuccessor(pm);
        pm.setSuccessor(dir);

        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the number of employees: ");
        int a= sc.nextInt();

        while(a!=0)
        {
            a=a-1;

            System.out.print("Enter leave type: ");
            String leavetype= sc.next();

            System.out.print("Enter employee name: ");
            String empname= sc.next();

            String leavestatus= "NEW";

            System.out.print("Enter request date: ");
            String rdate= sc.next();

            String leavereason="NotApplicable";
            String sdate="Not Applicable";
            String edate="Not Applicable";

            if(leavetype.equals("CL"))
            {
                System.out.print("Enter reason: ");
                leavereason= sc.next();
            }
            else if(leavetype.equals("VL"))
            {
                System.out.print("Enter start date : ");
                sdate= sc.next();
                System.out.print("Enter end date : ");
                edate= sc.next();
            }
            tlead.approve(new EmployeeLeave(leavetype,empname,leavestatus,rdate,leavereason,sdate,edate));
        }
```




```
52  }
53  class LeaveProcessorFactory {
54      public EmployeeLeaveProcessor createProcessor(String type) {
55          if(type.equals("TECHLEAD")) {
56              return new TechLead();
57          }
58          else if(type.equals("PROJECT MANAGER")) {
59              return new ProjectManager();
60          }
61          else if(type.equals("DIRECTOR")) {
62              return new Director();
63          }
64          return null;
65      }
66  }
67
68  class TechLead extends EmployeeLeaveProcessor
69  {
70      protected String getApprovalLimit(){
71          return "SL";
72      };
73      protected String getDesignation(){
74          return "TECHLEAD";
75      }
76  }
77  class ProjectManager extends EmployeeLeaveProcessor
78  {
79      protected String getApprovalLimit(){
80          return "CL";
81      };
82      protected String getDesignation(){
83          return "PROJECT MANAGER";
84      }
85  }
86  class Director extends EmployeeLeaveProcessor
87  {
88      protected String getApprovalLimit(){
89          return "VL";
90      };
91      protected String getDesignation(){
92          return "Director";
93      }
94  }
95  abstract class EmployeeLeaveProcessor{
96      EmployeeLeaveProcessor tp;
97      protected abstract String getApprovalLimit();
98      protected abstract String getDesignation();
99      void setSuccessor(EmployeeLeaveProcessor tp)
100     {
101         this.tp = tp;
102
103     }
104     void approve(EmployeeLeave t)
```

```
    {
        if((t.getleavetype()).equals(this.getApprovalLimit()))
        {
            t.approvedBy=this.getDesignation();
            t.leaveStatus="Done";
            t.approvalDate=t.requestDate;
            System.out.println("\n**********************\n");
            System.out.println(t.getleavetype()+" is approved by "+this.getDesignation());
            System.out.println("\nThe details of the employees are as follows:\n"+
        "Employee name : "+t.empName+"\nLeave Status : "+t.leaveStatus+"\nRequest Date : "+t.requestDate+"\nLeave Reason : "+t.leavereason
        +"\nStart Date : "+t.sdate+"\nEnd Date : "+t.edate);
            System.out.println("\n**********************\n");
        }
        else
        {
            tp.approve(t);
        }
    }
}

class EmployeeLeave{
    public String leavetype;
    public String empName;
    public String leaveStatus="New";
    public String approvedBy;
    public String requestDate;
    public String approvalDate;
    public String leavereason;
    public String sdate;
    public String edate;
    EmployeeLeave(String leavetype,String empName,String leaveStatus,String requestDate,String leavereason,String sdate,String edate)
    {
        this.leavetype=leavetype;
        this.empName=empName;
        this.leaveStatus=leaveStatus;
        this.requestDate=requestDate;
        this.leavereason=leavereason;
        this.sdate=sdate;
        this.edate=edate;
    }
    String getleavetype()
    {
        return leavetype;
    }
}
```

6. Input and Output Screenshots for all types of leaves

```
D:\PES1UG20CS588\SEM6\OOAD\Lab9&10>javac leave_man.java

D:\PES1UG20CS588\SEM6\OOAD\Lab9&10>java leave_man
Enter the number of employees: 3
Enter leave type: SL
Enter employee name: SAHANA
Enter request date: 17/01/2023

**********************

SL is approved by TECHLEAD

The details of the employees are as follows:
Employee name : SAHANA
Leave Status : Done
Request Date : 17/01/2023
Leave Reason : NotApplicable
Start Date : Not Applicable
End Date : Not Applicable

**********************
```


C:\Windows\System32\cmd.e
Start Date : Not Applicable
End Date : Not Applicable
****
Enter leave type: CL
Enter employee name: NIHARIKA
Enter request date: 18/05/2023
Enter reason: FEVER
****
CL is approved by PROJECT MANAGER
The details of the employees are as follows:
Employee name : NIHARIKA
Leave Status : Done
Request Date : 18/05/2023
Leave Reason : FEVER
Start Date : Not Applicable
End Date : Not Applicable
****
Enter leave type: VL
Enter employee name: SHREYA
Enter request date: 19/06/2023
Enter start date : 28/10/2023
Enter end date : 2/11/2023
****
VL is approved by Director
The details of the employees are as follows:
Employee name : SHREYA
Leave Status : Done
Request Date : 19/06/2023
Leave Reason : NotApplicable
Start Date : 28/10/2023
End Date : 2/11/2023
****
D:\PES1UG20CS588\SEM6\OOAD\Lab9&10>