OOAD LAB 9-10

Name : Sanmat Sanjayakumar Payagoudar

SRN : PES1UG20CS385

Section : G

Problem Statement :

A Company's Leave Management System has the following features. An Employee (client) can apply for Casual Leave (CL), Sick Leave (SL) and Vacation Leave (VL). The roles in the hierarchy who are responsible for approving or rejecting the leave using the process specified are Director, Project Manager and Tech Lead. The Leave request contains the following details: empName, leaveStatus, approvedBy, requestDate and approvalDate. A CL and SL are for only one day. A VL will have a startDate and endDate. A CL will also need a reason to be specified. The Leave created by the client is assigned a "New" status. If the leave is SL, then it will be processed by Tech Lead, if it is CL, it will be processed by the Project Manager, and if it is VL, will be processed by the Director. The Leave when created is sent to Tech Lead for processing, if it is not SL, the Tech Lead will just pass the request to the next higher level. Similarly, Project Manager will process a CL request or forward the VL request to the next higher level. Once the request is processed, a message should be displayed on the console showing request details and approval details. Represent the design (using appropriate design patterns) in a UML Class Diagram and implement the same.

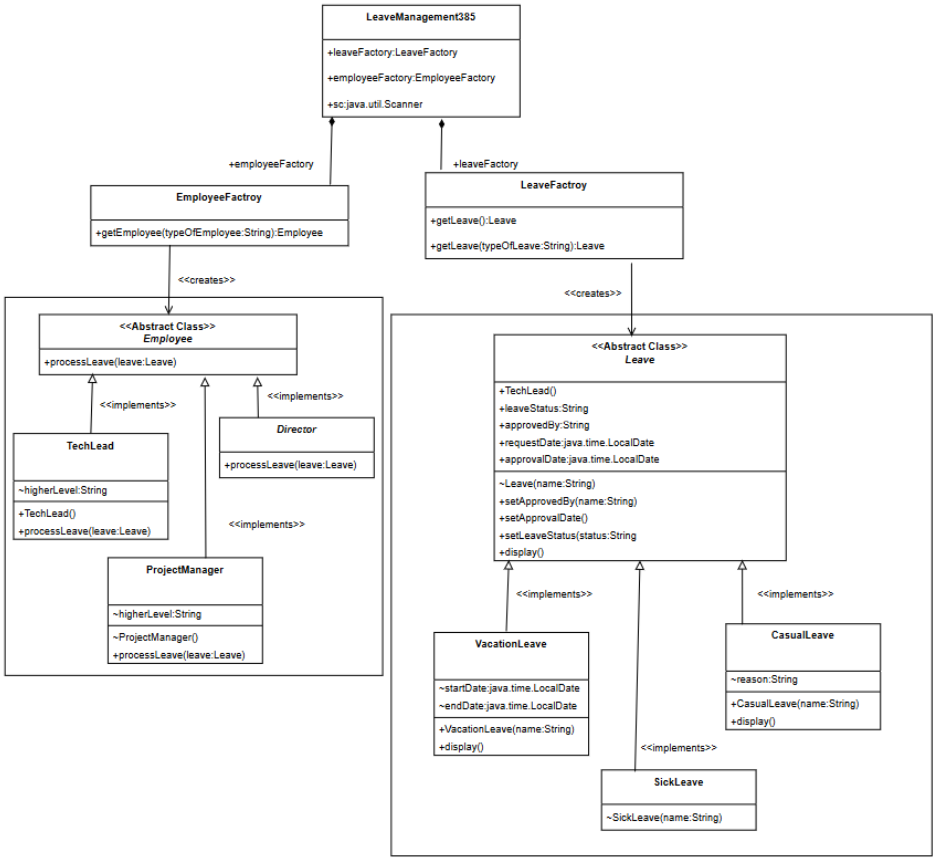
Design Patterns Considered:

- Abstract Factory pattern (Leave and Employee)
- Singleton Pattern (for Scanner and Factory classes)

Design Patterns Used:

- Abstract Factory pattern(Leave and Employee)

UML class model:



Code:

```
import java.util.Scanner;
import java.time.LocalDate;


public class LeaveManagement385 {


    public static LeaveFactory leaveFactory = new
LeaveFactory();
    public static EmployeeFactory employeeFactory =
new EmployeeFactory();
    public static Scanner sc = new
Scanner(System.in);


    public static void main(String[] args) {
        Leave leave = leaveFactory.getLeave();
        Employee e =
employeeFactory.getEmployee("TL");
        e.processLeave(leave);
        leave.display();
        sc.close();
        return;
    }
}


class LeaveFactory {
    public Leave getLeave() {
        System.out.println("\n<<< Hello! Fill the
details for leave approval >>>\n");
```

```
        System.out.println(">> Enter your name");
        String empName =
LeaveManagement385.sc.nextLine();
        System.out.println(">> Enter type of leave
(CL, SL, VL)");
        String typeOfLeave =
LeaveManagement385.sc.nextLine();
        if (typeOfLeave.equals("CL"))
            return new CasualLeave(empName);
        else if (typeOfLeave.equals("SL"))
            return new SickLeave(empName);
        else
            return new VacationLeave(empName);
    }
}


abstract class Leave {
    public String empname;
    public String leaveStatus;
    public String approvedBy;
    public LocalDate requestDate = LocalDate.now();
    public LocalDate approvalDate;


    Leave(String name) {
        this.empname = name;
    }


    public void setApprovedBy(String name) {
        this.approvedBy = name;
    }
```

```
    public void setApprovalDate() {
        this.approvalDate = LocalDate.now();
    }


    public void setLeaveStatus(String status) {
        this.leaveStatus = status;
    }


    public void display() {
        System.out.println("\n----Leave Report----
");
        System.out.println("empName: " +
this.empname);
        System.out.println("leaveStatus: " +
this.leaveStatus);
        System.out.println("requestDate: " +
this.requestDate);
        System.out.println("approvalDate: " +
this.approvalDate);
        System.out.println("approvedBy: " +
this.approvedBy);
    }
}


class SickLeave extends Leave {
    SickLeave(String name) {
        super(name);
    }
}
```

```
class CasualLeave extends Leave {
    String reason;


    public CasualLeave(String name) {
        super(name);
        System.out.println(">> Enter reason");
        this.reason =
LeaveManagement385.sc.nextLine();
    }


    @Override
    public void display() {
        super.display();
        System.out.println("Reason: " +
this.reason);
    }
}


class VacationLeave extends Leave {
    LocalDate startDate;
    LocalDate endDate;


    public VacationLeave(String name) {
        super(name);
        System.out.println(">> Enter start date");
        this.startDate =
LocalDate.parse(LeaveManagement385.sc.nextLine());
        System.out.println(">> Enter end date");
```

```
        this.endDate =
LocalDate.parse(LeaveManagement385.sc.nextLine());
    }


    @Override
    public void display() {
        super.display();
        System.out.println("Vacation Start date: "
+ this.startDate);
        System.out.println("vacation End date: " +
this.endDate);
    }
}


class EmployeeFactory {
    public Employee getEmployee(String
typeOfEmployee) {
        if (typeOfEmployee.equals("TL"))
            return new TechLead();
        else if (typeOfEmployee.equals("PM"))
            return new ProjectManager();
        else
            return new Director();
    }
}


abstract class Employee {
    public void processLeave(Leave leave) {
    };
}
```

```
class TechLead extends Employee {
    String higherLevel;


    public TechLead() {
        this.higherLevel = "PM";
    }


    @Override
    public void processLeave(Leave leave) {
        if (leave instanceof SickLeave) {
            leave.setApprovalDate();
            leave.setLeaveStatus("approved");
            leave.setApprovedBy("TechLead");
        } else {
            Employee projectManager =
LeaveManagement385.employeeFactory.getEmployee(high
erLevel);
            projectManager.processLeave(leave);
        }
    };
}


class ProjectManager extends Employee {
    String higherLevel;


    ProjectManager() {
        this.higherLevel = "Director";
    }
```

```
    @Override
    public void processLeave(Leave leave) {
        if (leave instanceof CasualLeave) {
            leave.setApprovalDate();
            leave.setLeaveStatus("approved");
            leave.setApprovedBy("ProjectManager");
        } else {
            Employee director =
LeaveManagement385.employeeFactory.getEmployee(high
erLevel);
            director.processLeave(leave);
        }
    };
}


class Director extends Employee {
    @Override
    public void processLeave(Leave leave) {
        if (leave instanceof VacationLeave) {
            leave.setApprovalDate();
            leave.setLeaveStatus("approved");
            leave.setApprovedBy("Director");
        } else {
            leave.setLeaveStatus("rejected");
        }
    };
}
```

I Input and Output Screenshots for all types of leaves

Casual Leave :

```
PS C:\Users\sanma\OneDrive\Documents> javac LeaveManagement385.java
PS C:\Users\sanma\OneDrive\Documents> java LeaveManagement385

<<< Hello! Fill the details for leave approval >>>

>> Enter your name
Sanmat
>> Enter type of leave (CL, SL, VL)
CL
>> Enter reason
Family Function

----Leave Report----
empName: Sanmat
leaveStatus: approved
requestDate: 2023-04-18
approvalDate: 2023-04-18
approvedBy: ProjectManager
Reason: Family Function
PS C:\Users\sanma\OneDrive\Documents>
```

Sick Leave :

```
PS C:\Users\sanma\OneDrive\Documents> java LeaveManagement385

<<< Hello! Fill the details for leave approval >>>

>> Enter your name
Sanmat
>> Enter type of leave (CL, SL, VL)
SL

----Leave Report----
empName: Sanmat
leaveStatus: approved
requestDate: 2023-04-18
approvalDate: 2023-04-18
approvedBy: TechLead
PS C:\Users\sanma\OneDrive\Documents>
```

Vacation Leave :