**Name: Adithya M SRN: PES1UG20CS621 Section: K**

1. **Problem statement**

* The system is able to handle different types of leave requests, such as sick leave, casual leave, and vacation leave, and route them to the appropriate manager for approval.
* The system also keeps track of the status of each request and notify the employee when their request has been approved or denied.

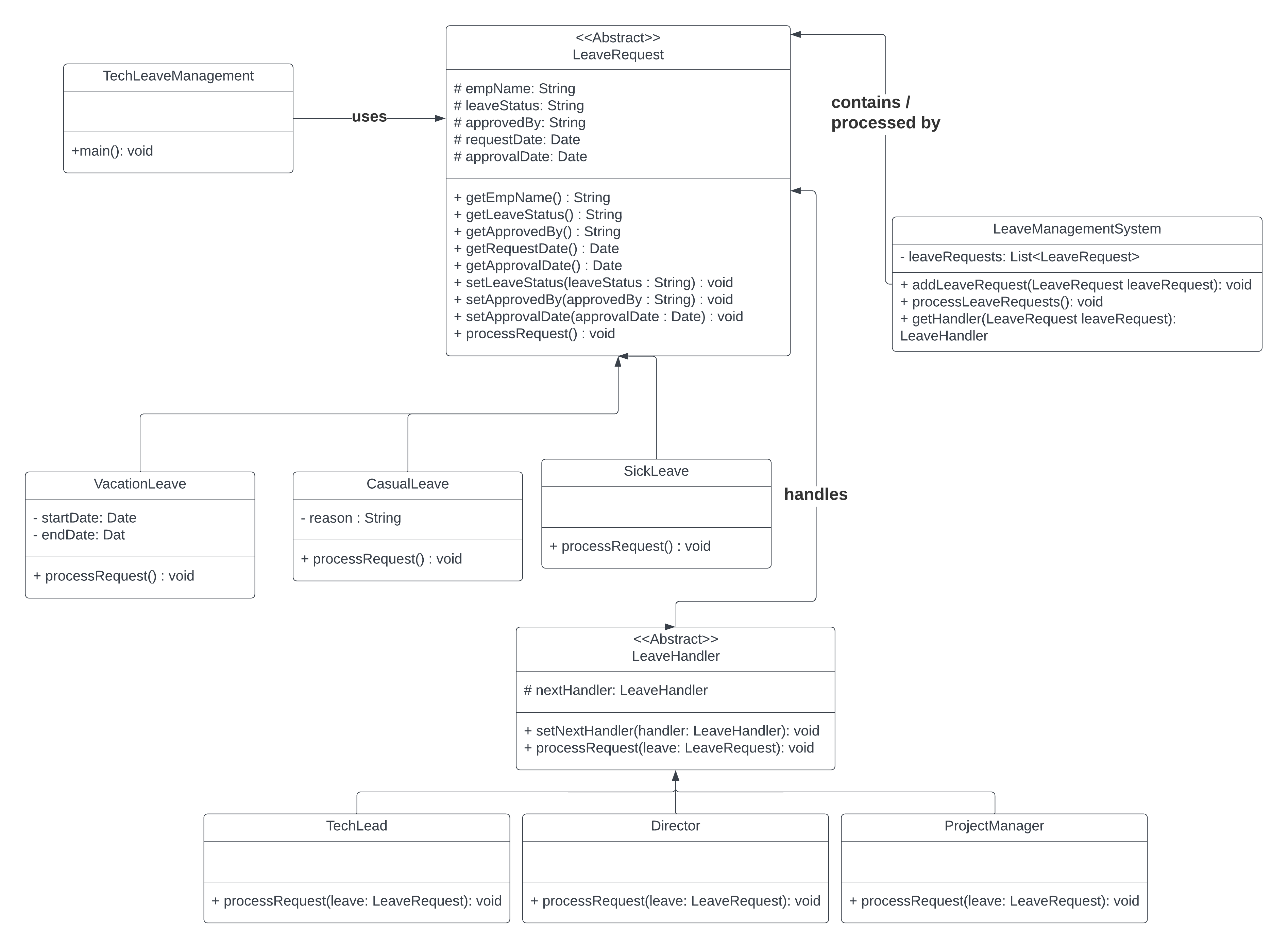
1. **Design patterns considered:**

* Chain of Responsibility pattern
* Factory Method pattern

1. **Design patterns used:**

* Chain of Responsibility pattern as this pattern allows us to create a chain of handlers that can process leave requests in a specific order. Each handler in the chain can either approve the request or pass it on to the next handler in the chain.
* This pattern also allows for better extensibility as its easier to add more Employees and types of leaves by implementing the abstract class.

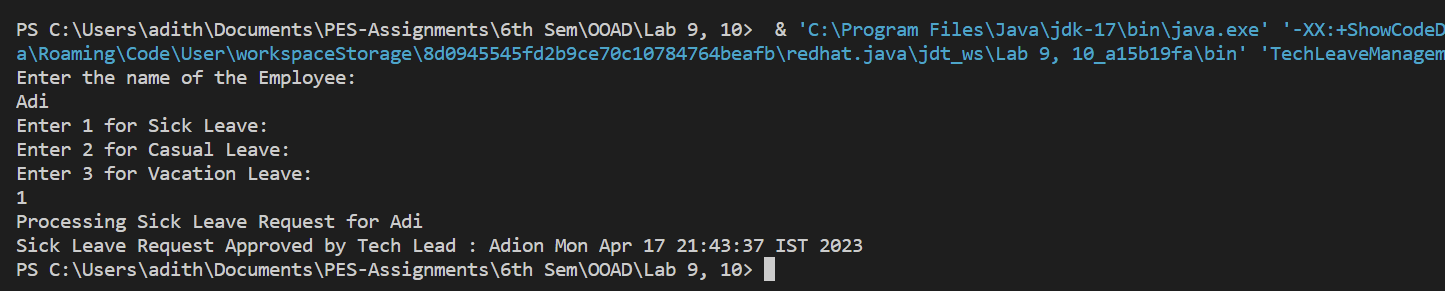
1. **UML Class Model:**

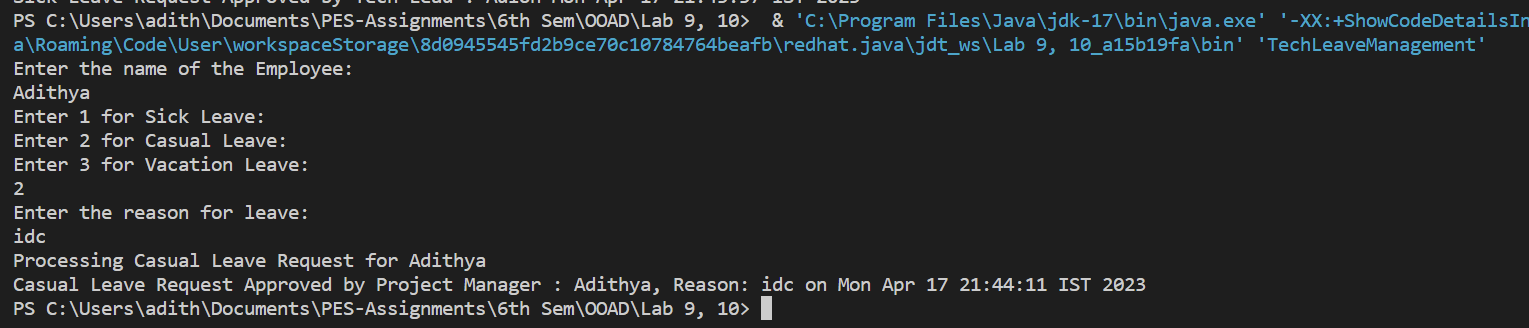


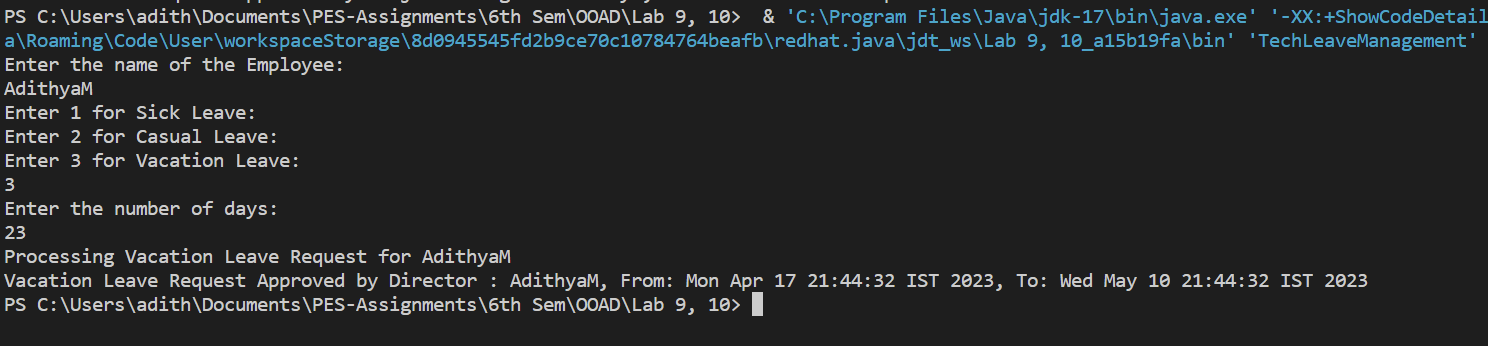
**5. Code:**

|  |
| --- |
| import java.util.Date;  import java.util.Scanner;  public class TechLeaveManagement {  public static void main(String[] args) {  LeaveHandler techLead = new TechLead();  LeaveHandler projectManager = new ProjectManager();  LeaveHandler director = new Director();  techLead.setNextHandler(projectManager);  projectManager.setNextHandler(director);  Scanner sc = new Scanner(System.in);  System.out.println("Enter the name of the Employee: ");  String ename = sc.next();  System.out.println("Enter 1 for Sick Leave: ");  System.out.println("Enter 2 for Casual Leave: ");  System.out.println("Enter 3 for Vacation Leave: ");  int leave = sc.nextInt();  switch (leave) {  case 1:  System.out.println("Processing Sick Leave Request for " + ename);  LeaveRequest sickLeave = new SickLeave(ename);  techLead.processRequest(sickLeave);  break;  case 2:  System.out.println("Enter the reason for leave: ");  String r = sc.next();  System.out.println("Processing Casual Leave Request for " + ename);  LeaveRequest casualLeave = new CasualLeave(ename, r);  techLead.processRequest(casualLeave);  break;  case 3:  System.out.println("Enter the number of days: ");  int nod = sc.nextInt();  long nod1 = nod \* 86400000;  System.out.println("Processing Vacation Leave Request for " + ename);  LeaveRequest vacationLeave = new VacationLeave(ename, new Date(),  new Date(System.currentTimeMillis() + nod1));  techLead.processRequest(vacationLeave);  break;  default:  System.out.println("Invalid Choice");  }  sc.close();  }  } |
| import java.util.Date;  abstract class LeaveRequest {  protected String empName;  protected String leaveStatus;  protected String approvedBy;  protected Date requestDate;  protected Date approvalDate;  public LeaveRequest(String empName) {  this.empName = empName;  this.leaveStatus = "New";  this.requestDate = new Date();  }  public String getEmpName() {  return empName;  }  public String getLeaveStatus() {  return leaveStatus;  }  public String getApprovedBy() {  return approvedBy;  }  public Date getRequestDate() {  return requestDate;  }  public Date getApprovalDate() {  return approvalDate;  }  public void setLeaveStatus(String leaveStatus) {  this.leaveStatus = leaveStatus;  }  public void setApprovedBy(String approvedBy) {  this.approvedBy = approvedBy;  }  public void setApprovalDate(Date approvalDate) {  this.approvalDate = approvalDate;  }  public abstract void processRequest();  } |
| import java.util.ArrayList;  import java.util.List;  public class LeaveManagementSystem {  private List<LeaveRequest> leaveRequests;  public LeaveManagementSystem() {  leaveRequests = new ArrayList<>();  }  public void addLeaveRequest(LeaveRequest leaveRequest) {  leaveRequests.add(leaveRequest);  }  public void processLeaveRequests() {  for (LeaveRequest leaveRequest : leaveRequests) {  LeaveHandler handler = getHandler(leaveRequest);  if (handler != null) {  handler.processRequest(leaveRequest);  System.out.println("Leave request approved: " + leaveRequest);  }  else {  System.out.println("No handler found for leave request: " + leaveRequest);  }  }  }  private LeaveHandler getHandler(LeaveRequest leaveRequest) {  if (leaveRequest instanceof SickLeave) {  return new TechLead();  } else if (leaveRequest instanceof CasualLeave) {  return new ProjectManager();  } else if (leaveRequest instanceof VacationLeave) {  return new Director();  } else {  return null;  }  }  } |
| abstract class LeaveHandler {  protected LeaveHandler nextHandler;  public void setNextHandler(LeaveHandler handler) {  this.nextHandler = handler;  }  public abstract void processRequest(LeaveRequest leave);  } |
| import java.util.Date;  class CasualLeave extends LeaveRequest {    private String reason;  public CasualLeave(String empName, String reason) {  super(empName);  this.reason = reason;  }    public void processRequest() {  this.leaveStatus = "Approved";  this.approvedBy = "Project Manager";  this.approvalDate = new Date(System.currentTimeMillis());  System.out.println("Casual Leave Request Approved by " + this.approvedBy + " : " + this.empName + ", Reason: "  + this.reason + " on " + this.approvalDate);  }  } |
| import java.util.Date;  class SickLeave extends LeaveRequest {  public SickLeave(String empName) {  super(empName);  }  public void processRequest() {  this.leaveStatus = "Approved";  this.approvedBy = "Tech Lead";  this.approvalDate = new Date(System.currentTimeMillis());  System.out.println(  "Sick Leave Request Approved by " + this.approvedBy + " : " + this.empName + "on " + this.approvalDate);  }  } |
| import java.util.Date;  class VacationLeave extends LeaveRequest {  private Date startDate;  private Date endDate;  public VacationLeave(String empName, Date startDate, Date endDate) {  super(empName);  this.startDate = startDate;  this.endDate = endDate;  }    public void processRequest() {  this.leaveStatus = "Approved";  this.approvedBy = "Director";  this.approvalDate = new Date();  System.out.println("Vacation Leave Request Approved by " + this.approvedBy + " : " + this.empName + ", From: "  + this.startDate + ", To: " + this.endDate);  }  } |
| class TechLead extends LeaveHandler {    public void processRequest(LeaveRequest leave) {  if (leave instanceof SickLeave) {  leave.processRequest();  } else if (nextHandler != null) {  nextHandler.processRequest(leave);  }  }  } |
| class ProjectManager extends LeaveHandler {  public void processRequest(LeaveRequest leave) {  if (leave instanceof CasualLeave) {  leave.processRequest();  } else if (nextHandler != null) {  nextHandler.processRequest(leave);  }  }  } |
| class Director extends LeaveHandler {  public void processRequest(LeaveRequest leave) {  if (leave instanceof VacationLeave) {  leave.processRequest();  } else if (nextHandler != null) {  nextHandler.processRequest(leave);  }  }  } |

1. **Input and Output Screenshots for all types of leaves:**

****

****

****