

UE20CS352-OOADJ Lab-9&10

NAME	SRN	CLASS & SECTION
Vijay J	PES2UG20CS815	6 - J

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 processes 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.

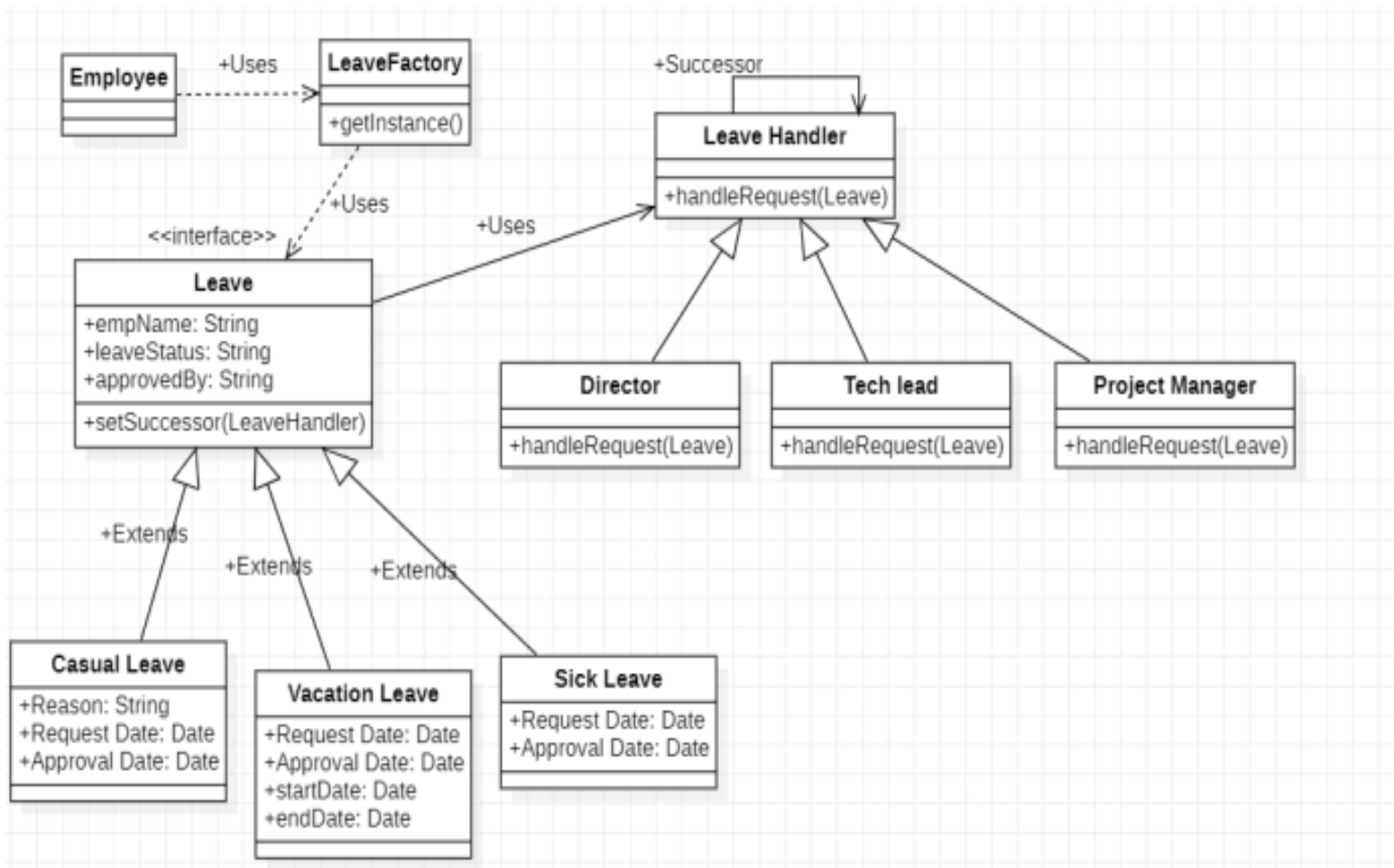
Design patterns considered:

- Chain of Responsibility Pattern.
- Facade Method
- Builder Method
- Factory Method

Design patterns used:

- Chain of Responsibility Pattern.
- Factory Method

UML Class Model:



CODE:

PES2UG20CS815.java

```
Open PES2UG20CS815.java Save - □ ×
~/PESU/6th Sem/OOAD/OOAD LAB/Week - 9&10

1
2 // This class represents the main class that creates and tests the chain of
  responsibility
3 import java.util.Scanner; // for user input
4
5 public class PES2UG20CS815 {
6
7     public static void main(String[] args) {
8
9         // Create a chain of handlers: Tech Lead → Project Manager → Director
10        Handler techLead = new TechLead(new ProjectManager(new Director()));
11
12        // Create a scanner object for user input
13        Scanner scanner = new Scanner(System.in);
14
15        // Ask the user for the number of leave requests to test
16        System.out.println("How many leave requests do you want to test?");
17        int n = scanner.nextInt(); // read an integer from the user
18
19        // Loop n times to create and test n leave requests
20        for (int i = 0; i < n; i++) {
21
22            // Ask the user for the details of the leave request
23            System.out.println("Enter the employee name:");
24            String empName = scanner.next(); // read a string from the user
25            System.out.println("Enter the leave type (SL, CL, or VL):");
26            String leaveType = scanner.next(); // read a string from the user
27            System.out.println("Enter the request date (dd/mm/yyyy):");
28            String requestDate = scanner.next(); // read a string from the user
29
30            // Create a leave request object based on the leave type
31            LeaveRequest request;
32
33            if (leaveType.equals("SL") || leaveType.equals("CL")) {
34                // Ask the user for the reason for SL or CL
35                System.out.println("Enter the reason for " + leaveType + ":");
36                String reason = scanner.next(); // read a string from the user
37                // Create a SL or CL request with the reason
38                request = new LeaveRequest(empName, leaveType, requestDate, reason);
39            } else if (leaveType.equals("VL")) {
40                // Ask the user for the start date and end date for VL
41                System.out.println("Enter the start date (dd/mm/yyyy) for VL:");
42                String startDate = scanner.next(); // read a string from the user
43                System.out.println("Enter the end date (dd/mm/yyyy) for VL:");
44                String endDate = scanner.next(); // read a string from the user
45                // Create a VL request with the start date and end date
46                request = new LeaveRequest(empName, leaveType, requestDate,
47                startDate, endDate);
48            } else {
49                // Invalid leave type entered by the user
50                System.out.println("Error: Invalid leave type entered");
51                continue; // skip this iteration and go to the next one
52            }
53
54            // Test the chain of responsibility with the leave request
55            techLead.handle(request);
56
57            // Close the scanner object
58            scanner.close();
59        }
60    }
61 }
```

Handler.java

```
Handler.java
~/PESU/6th Sem/OOAD/OOAD LAB/Week - 9&10
Save

1 // This interface represents a handler in the chain of responsibility
2 public interface Handler {
3
4     // This method handles the leave request or passes it to the next handler
5     public void handle(LeaveRequest request);
6 }
```

Director.java

```
Director.java
~/PESU/6th Sem/OOAD/OOAD LAB/Week - 9&10
Save

1 // This class represents a Director handler in the chain of responsibility
2 public class Director implements Handler {
3
4     // The constructor that does not take any parameter
5     public Director() {
6         // No need to set the next handler as this is the last handler in the chain
7     }
8
9     @Override
10    public void handle(LeaveRequest request) {
11
12        System.out.println("Director received the leave request from " +
13        request.getEmpName());
14
15        // If the leave type is VL, process it and approve or reject it based on the
16        // duration
17        if (request.getLeaveType().equals("VL")) {
18            System.out.println("Director processed the VL request");
19            // Assuming some logic to calculate the duration and decide whether to
20            // approve
21            // or reject
22            int duration = calculateDuration(request.getStartDate(),
23            request.getEndDate());
24            if (duration <= 10) { // assuming 10 days is the maximum allowed VL
25                duration
26                System.out.println("Director approved the VL request");
27                request.setLeaveStatus("Approved");
28                request.setApprovedBy("Director");
29                request.setApprovalDate("Today"); // assuming today's date
30            } else {
31                System.out.println("Director rejected the VL request");
32                request.setLeaveStatus("Rejected");
33            }
34        }
35    }
36 }
```

```

29         request.setApprovedBy("Director");
30         request.setApprovalDate("Today"); // assuming today's date
31     }
32 }
33 // Otherwise, print an error message as there is no more handler in the chain
34 else {
35     System.out.println("Error: Invalid leave type for Director handler");
36 }
37 }
38
39 // A helper method to calculate the duration of VL in days
40 private int calculateDuration(String startDate, String endDate) {
41     // Assuming some logic to parse and compare the dates and return the
42     // difference
43     // in days
44     return 5; // for simplicity, return 5 for now
45 }

```

TechLead.java

```

TechLead.java
~/PESU/6th Sem/OOAD/OOAD LAB/Week - 9&10
Save

1 // This class represents a Tech Lead handler in the chain of responsibility
2 public class TechLead implements Handler {
3
4     // The next handler in the chain
5     private Handler next;
6
7     // The constructor that takes the next handler as a parameter
8     public TechLead(Handler next) {
9         this.next = next;
10    }
11
12    @Override
13    public void handle(LeaveRequest request) {
14
15        System.out.println("Tech Lead received the leave request from " +
16        request.getEmpName());
17
18        // If the leave type is SL, process it and approve it
19        if (request.getLeaveType().equals("SL")) {
20            System.out.println("Tech Lead approved the SL request");
21            request.setLeaveStatus("Approved");
22            request.setApprovedBy("Tech Lead");
23            request.setApprovalDate("Today"); // assuming today's date
24        }
25        // Otherwise, forward it to the next handler
26        else {
27            System.out.println("Tech Lead forwarded the request to the next handler");
28            next.handle(request);
29        }
30    }
31 }

```

ProjectManager.java

```
ProjectManager.java
~/PESU/6th Sem/OOAD/OOAD LAB/Week - 9&10
Save

1 // This class represents a Project Manager handler in the chain of responsibility
2 public class ProjectManager implements Handler {
3
4     // The next handler in the chain
5     private Handler next;
6
7     // The constructor that takes the next handler as a parameter
8     public ProjectManager(Handler next) {
9         this.next = next;
10    }
11
12    @Override
13    public void handle(LeaveRequest request) {
14
15        System.out.println("Project Manager received the leave request from " +
16            request.getEmpName());
17
18        // If the leave type is CL, process it and approve or reject it based on the
19        // reason
20        if (request.getLeaveType().equals("CL")) {
21            System.out.println("Project Manager processed the CL request");
22            // Assuming some logic to check the reason and decide whether to approve
23            or
24            // reject
25            boolean validReason = checkReason(request.getReason());
26            if (validReason) {
27                System.out.println("Project Manager approved the CL request");
28                request.setLeaveStatus("Approved");
29                request.setApprovedBy("Project Manager");
30                request.setApprovalDate("Today"); // assuming today's date
31            } else {
32                System.out.println("Project Manager rejected the CL request");
33                request.setLeaveStatus("Rejected");
34                request.setApprovedBy("Project Manager");
35                request.setApprovalDate("Today"); // assuming today's date
36            }
37        }
38        // Otherwise, forward it to the next handler
39        else {
40            System.out.println("Project Manager forwarded the request to the next
41            handler");
42            next.handle(request);
43        }
44
45        // A helper method to check the reason for CL and return true or false
46        private boolean checkReason(String reason) {
47            // Assuming some logic to validate the reason
48            return true; // for simplicity, return true for now
49        }
50    }
51 }
```

LeaveRequest.java

```
Open  ▾  [Icon]  LeaveRequest.java  Save  [Menu Icon]  -  [Maximize Icon]  [Close Icon]
~ /PESU/6th Sem/OOAD/OOAD LAB/Week - 9&10

1 // This class represents a leave request with the details
2 public class LeaveRequest {
3     // The employee name
4     private String empName;
5     // The leave type: SL, CL, or VL
6     private String leaveType;
7     // The leave status: New, Approved, or Rejected
8     private String leaveStatus;
9     // The name of the handler who approved or rejected the request
10    private String approvedBy;
11    // The date of the request
12    private String requestDate;
13    // The date of the approval or rejection
14    private String approvalDate;
15    // The start date of the VL
16    private String startDate;
17    // The end date of the VL
18    private String endDate;
19    // The reason for the CL
20    private String reason;
21
22    // Constructor for SL and CL requests
23    public LeaveRequest(String empName, String leaveType, String requestDate, String
reason) {
24        this.empName = empName;
25        this.leaveType = leaveType;
26        this.requestDate = requestDate;
27        this.reason = reason;
28        this.leaveStatus = "New"; // default status
29        this.approvedBy = null; // default value
30        this.approvalDate = null; // default value
31        this.startDate = null; // not applicable
32        this.endDate = null; // not applicable
33    }
34
35    // Constructor for VL requests
36    public LeaveRequest(String empName, String leaveType, String requestDate, String
startDate, String endDate) {
37        this.empName = empName;
38        this.leaveType = leaveType;
39        this.requestDate = requestDate;
40        this.startDate = startDate;
41        this.endDate = endDate;
42        this.leaveStatus = "New"; // default status
43        this.approvedBy = null; // default value
44        this.approvalDate = null; // default value
45        this.reason = null; // not applicable
46    }
47
48    // Getters and setters for the fields
49
50    public String getEmpName() {
51        return empName;
52    }
53
54    public void setEmpName(String empName) {
55        this.empName = empName;
56    }
57
58    public String getLeaveType() {
59        return leaveType;
60    }
61
62    public void setLeaveType(String leaveType) {
```

```
63     this.leaveType = leaveType;
64 }
65
66 public String getLeaveStatus() {
67     return leaveStatus;
68 }
69
70 public void setLeaveStatus(String leaveStatus) {
71     this.leaveStatus = leaveStatus;
72 }
73
74 public String getApprovedBy() {
75     return approvedBy;
76 }
77
78 public void setApprovedBy(String approvedBy) {
79     this.approvedBy = approvedBy;
80 }
81
82 public String getRequestDate() {
83     return requestDate;
84 }
85
86 public void setRequestDate(String requestDate) {
87     this.requestDate = requestDate;
88 }
89
90 public String getApprovalDate() {
91     return approvalDate;
92 }
93
94 public void setApprovalDate(String approvalDate) {
```

```
95     this.approvalDate = approvalDate;
96 }
97
98 public String getStartDate() {
99     return startDate;
100 }
101
102 public void setStartDate(String startDate) {
103     this.startDate = startDate;
104 }
105
106 public String getEndDate() {
107     return endDate;
108 }
109
110 public void setEndDate(String endDate) {
111     this.endDate = endDate;
112 }
113
114 public String getReason() {
115     return reason;
116 }
117
118 public void setReason(String reason) {
119     this.reason = reason;
120 }
121 }
```


OUTPUT:

Sick Leave (SL):

```
~/P/6/O/O/Week - 9&10

yoyo@zaemon in ~/PESU/6th Sem/OOAD/OOAD LAB/Week - 9&10 via  v17.0.7 took 41ms
λ javac PES2UG20CS815.java

yoyo@zaemon in ~/PESU/6th Sem/OOAD/OOAD LAB/Week - 9&10 via  v17.0.7 took 3s
λ java PES2UG20CS815
How many leave requests do you want to test?
1
Enter the employee name:
vijay
Enter the leave type (SL, CL, or VL):
SL
Enter the request date (dd/mm/yyyy):
28/04/2023
Enter the reason for SL:
Fever and Cough
Tech Lead received the leave request from vijay
Tech Lead approved the SL request

yoyo@zaemon in ~/PESU/6th Sem/OOAD/OOAD LAB/Week - 9&10 via  v17.0.7 took 1m46s
λ _
```

Casual Leave (CL):

```
~/P/6/O/O/Week - 9&10

yoyo@zaemon in ~/PESU/6th Sem/OOAD/OOAD LAB/Week - 9&10 via  v17.0.7 took 45ms
λ java PES2UG20CS815
How many leave requests do you want to test?
1
Enter the employee name:
chandan
Enter the leave type (SL, CL, or VL):
CL
Enter the request date (dd/mm/yyyy):
29/04/2023
Enter the reason for CL:
Family Problem
Tech Lead received the leave request from chandan
Tech Lead forwarded the request to the next handler
Project Manager received the leave request from chandan
Project Manager processed the CL request
Project Manager approved the CL request

yoyo@zaemon in ~/PESU/6th Sem/OOAD/OOAD LAB/Week - 9&10 via  v17.0.7 took 1m28s
λ _
```

Vacation Leave (VL):

```
~/P/6/O/O/Week - 9&10
yoyo@zaemon in ~/PESU/6th Sem/00AD/00AD LAB/Week - 9&10 via v17.0.7 took 47ms
λ java PES2UG20CS815
How many leave requests do you want to test?
1
Enter the employee name:
chinmay
Enter the leave type (SL, CL, or VL):
VL
Enter the request date (dd/mm/yyyy):
28/04/2023
Enter the start date (dd/mm/yyyy) for VL:
02/05/2023
Enter the end date (dd/mm/yyyy) for VL:
12/05/2023
Tech Lead received the leave request from chinmay
Tech Lead forwarded the request to the next handler
Project Manager received the leave request from chinmay
Project Manager forwarded the request to the next handler
Director received the leave request from chinmay
Director processed the VL request
Director approved the VL request

yoyo@zaemon in ~/PESU/6th Sem/00AD/00AD LAB/Week - 9&10 via v17.0.7 took 37s
λ
```