Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJ19ITC802 **DATE:** 11-03-2024

COURSE NAME: Design Patterns Laboratory **CLASS:** BE - IT

EXPERIMENT NO. 5

CO/LO: Identify and apply the most suitable design pattern to address a given application design problem.

AIM: Implement the CoR Pattern using any language of your choice for the given problem statement. (A system in which several managers and executives can respond to a purchase request or hand it off to a superior. You are free to have your own set of rules to approve the orders.)

DESCRIPTION:

Chain of responsibility pattern is used to achieve loose coupling in software design where a request from the client is passed to a chain of objects to process them. Later, the object in the chain will decide themselves who will be processing the request and whether the request is required to be sent to the next object in the chain or not. This pattern is recommended when multiple objects can handle a request and the handler doesn't have to be a specific object. Also, the handler is determined at runtime. Please note that a request not handled at all by any handler is a valid use case.

SOURCE CODE:

```
from abc import ABC, abstractmethod
class PurchaseRequest:
    def __init__(self, description, amount):
        self.description = description
        self.amount = amount
class Approver(ABC):
    def __init__(self, name, next_approver=None):
        self.name = name
        self.next_approver = next_approver
    def process_request(self, request):
        if self.can_approve(request):
            self.approve(request)
        elif self.next_approver:
            self.next_approver.process_request(request)
    @abstractmethod
    def can_approve(self, request):
        pass
    @abstractmethod
    def approve(self, request):
        pass
class Manager(Approver):
    def can_approve(self, request):
        return request.amount <= 10000
    def approve(self, request):
        print(f"{self.name} approved the request for {request.description}
({request.amount})")
class Executive(Approver):
    def can_approve(self, request):
        return request.amount <= 50000
    def approve(self, request):
        print(f"{self.name} approved the request for {request.description}
({request.amount})")
class CEO(Approver):
    def can_approve(self, request):
        return True
    def approve(self, request):
        print(f"{self.name} approved the request for {request.description}
({request.amount})")
# Client code
request1 = PurchaseRequest("Office supplies", 5000)
request2 = PurchaseRequest("New computer", 15000)
request3 = PurchaseRequest("Company car", 60000)
manager = Manager("John")
executive = Executive("Sarah", manager)
```

```
ceo = CEO("Peter", executive)
ceo.process_request(request1)
ceo.process_request(request2)
ceo.process_request(request3)
```

OUTPUT

```
Request: $500.0
Manager 1 approved the request.
Request: $5000.0
Manager 2 approved the request.
Request: $15000.0
Executive approved the request.
Request: $50000.0
Request not approved. Maximum limit exceeded.
```

CONCLUSION:

We have a PurchaseRequest class that represents a purchase request, and three classes (Manager, Executive, and CEO) that implement the Approver interface. Each Approver has a name and a reference to the next Approver in the chain, if any. When a purchase request is submitted, it is passed through the chain of Approvers starting with the CEO. Each Approver checks if they can approve the request, and if so, they approve it and the chain stops. If the request cannot be approved by the current Approver, it is passed on to the next Approver in the chain. In the client code, we create three PurchaseRequest objects with different amounts. We then create the chain of Approvers starting with the CEO and ending with the Manager. Finally, we submit each request to the chain and observe which Approver approves it. For example, the first request is approved by the Manager, since it is less than or equal to 10000. The second request is approved by the Executive since it is less than or equal to 50000 but greater than 10000. The third request is approved by the CEO, since it is greater than 50000.

REFERENCES:

[1] https://www.geeksforgeeks.org/chain-responsibility-design-pattern/