Assignment 9
Identification and Implementation of GOF pattern

AIM:

Apply any four GOF patterns to refine Design Model for a given problem description using effective UML 2 diagrams and implement them with a suitable object-oriented language.

PROBLEM STATEMENT:

Identification and Implementation of GOF pattern
Apply any two GOF pattern to refine the Design Model for a given problem description using effective UML 2 diagrams and implement them with a suitable object-oriented language

OBJECTIVE:

- To Study GOF patterns.
- To identify the applicability of GOF in the system.
- Implement a system with patterns.

1. RELEVANT THEORY:

The GoF Design Patterns are broken into three categories: Creational Patterns for the creation of objects; Structural Patterns to provide a relationship between objects; and finally, Behavioral Patterns to help define how objects interact.

Creational Design Patterns

- **Abstract Factory.** Allows the creation of objects without specifying their concrete type.
- **Builder.** Uses to create complex objects.
- **Factory Method.** Creates objects without specifying the exact class to create.
- **Prototype.** Creates a new object from an existing object.
- **Singleton.** Ensures only one instance of an object is created.

Structural Design Patterns

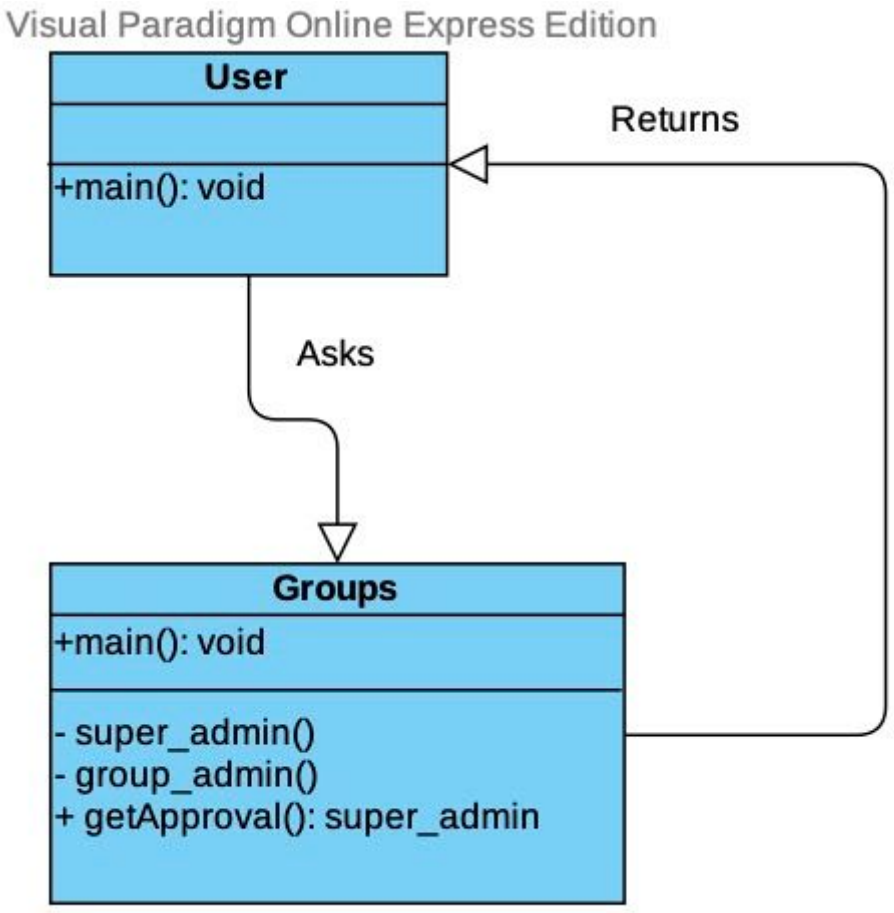
- **Adapter.** Allows for two incompatible classes to work together by wrapping an interface around one of the existing classes.
- **Bridge.** Decouples an abstraction so two classes can vary independently.
- **Composite.** Takes a group of objects into a single object.
- **Decorator.** Allows for an object's behaviour to be extended dynamically at run time.
- **Facade.** Provides a simple interface to a more complex underlying object.
- **Flyweight.** Reduces the cost of complex object models.
- **Proxy.** Provides a placeholder interface to an underlying object to control access, reduce cost, or reduce complexity.

Behavior Design Patterns

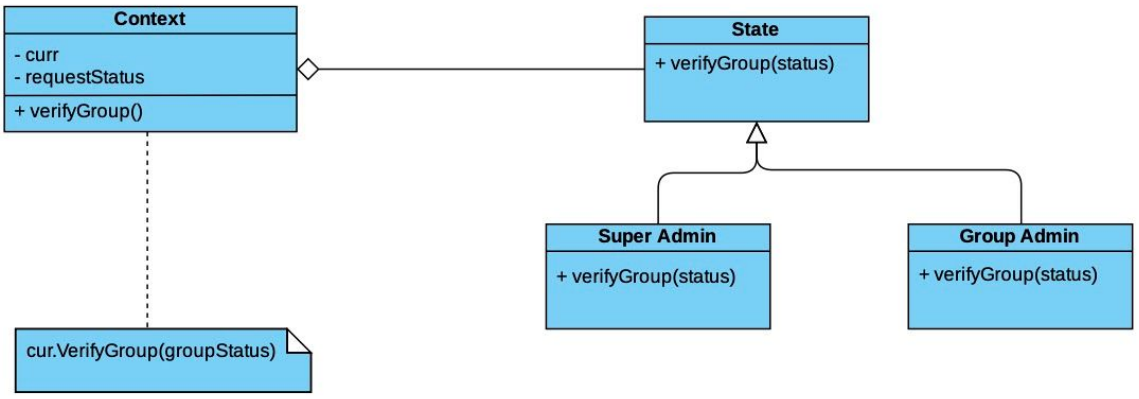
- **Chain of Responsibility.** Delegates command to a chain of processing objects.
- **Command.** Creates objects which encapsulate actions and parameters.
- **Interpreter.** Implements a specialized language.
- **Iterator.** Accesses the elements of an object sequentially without exposing its underlying representation.
- **Mediator.** Allows loose coupling between classes by being the only class that has detailed knowledge of their methods.
- **Memento.** Provides the ability to restore an object to its previous state.
- **Observer.** Is a publish/subscribe pattern which allows a number of observer objects to see an event.
- **State.** Allows an object to alter its behaviour when its internal state changes.
- **Strategy.** Allows one of a family of algorithms to be selected on-the-fly at run-time.
- **Template Method.** Defines the skeleton of an algorithm as an abstract class, allowing its subclasses to provide concrete behaviour.
- **Visitor.** Separates an algorithm from an object structure by moving the hierarchy of methods into one object.

2. IMPLEMENTATION:

- **Singleton -**
  The singleton pattern is one of the Gang of Four creational design patterns. In software engineering, this is a term that implies a class which can only be instantiated once, and a global point of access to that instance is provided. In the Java programming language, there are a few standard implementations of the singleton pattern. Here are some common ways of implementing a singleton.



- **State Design Pattern -**
  State pattern is one of the behavioral design patterns. State design pattern is used when an Object changes its behavior based on its internal state. If we have to change behavior of an object based on its state, we can have a state variable in the Object and use if-else condition block to perform different actions based on the state. State pattern is used to provide a systematic and lose-coupled way to achieve this through Context and State implementations.



CONCLUSION:

Thus in this assignment we have successfully identified and implemented GOF patterns for College Event Planning.