SDET Course

Design Patterns - Observer

3 Types of Design Patterns

- Creational
  - Singleton
  - Builder
  - Prototype
  - Factory Method
  - Abstract Factory

- Structural
  - Adapter
  - Composite
  - Proxy
  - Flyweight
  - Bridge
  - Facade
  - Decorator

- Behavioral
  - Strategy
  - **Observer**
  - Command
  - Memento
  - State
  - Template Method
  - Mediator
  - Chain of Responsibility
  - Interpreter
  - Visitor
  - Iterator

Agenda

- Description
- Diagram
- Code sample (Java)
- Use cases

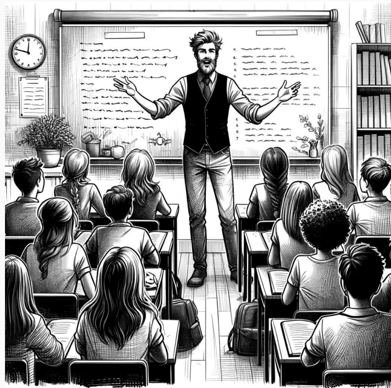
Descrip

on

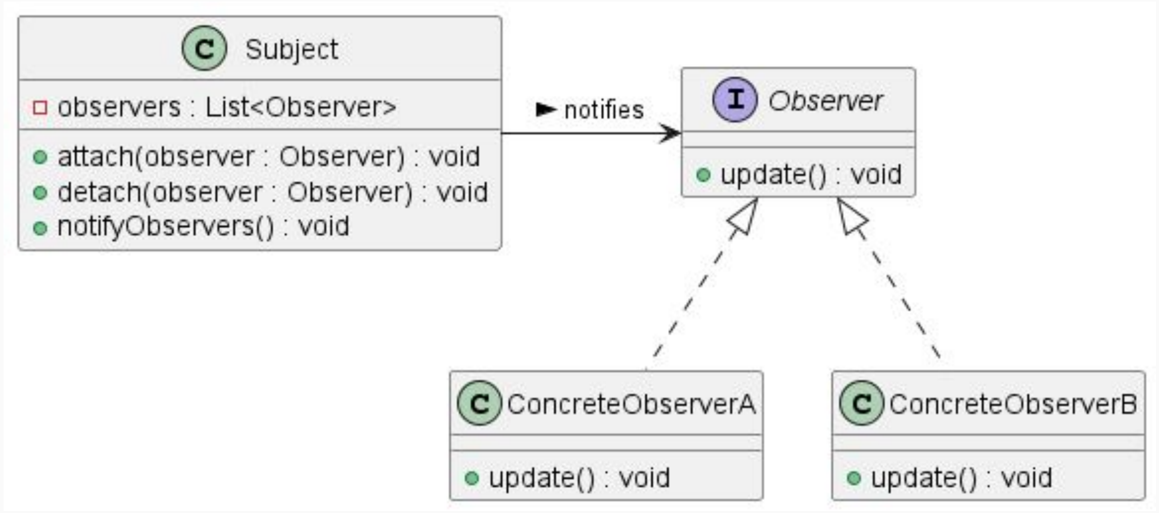
Introduction - The Broadcasting Teacher

Description

The Observer design pattern is a behavioral design pattern that focuses on establishing a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically. This pattern is primarily used to implement distributed event handling systems, in which the object that emits the event is the "subject", and the objects that receive the event are "observers". The Observer pattern allows for the objects to remain loosely coupled, whereby the subject doesn't need to know anything about the observers, other than that they implement a certain interface. This design principle enhances flexibility and reusability, making it easier to add new observers without modifying the subject or other observers. This pattern is widely used in various programming environments for implementing event management systems, such as graphical user interfaces where actions like button clicks are handled in an efficient manner.

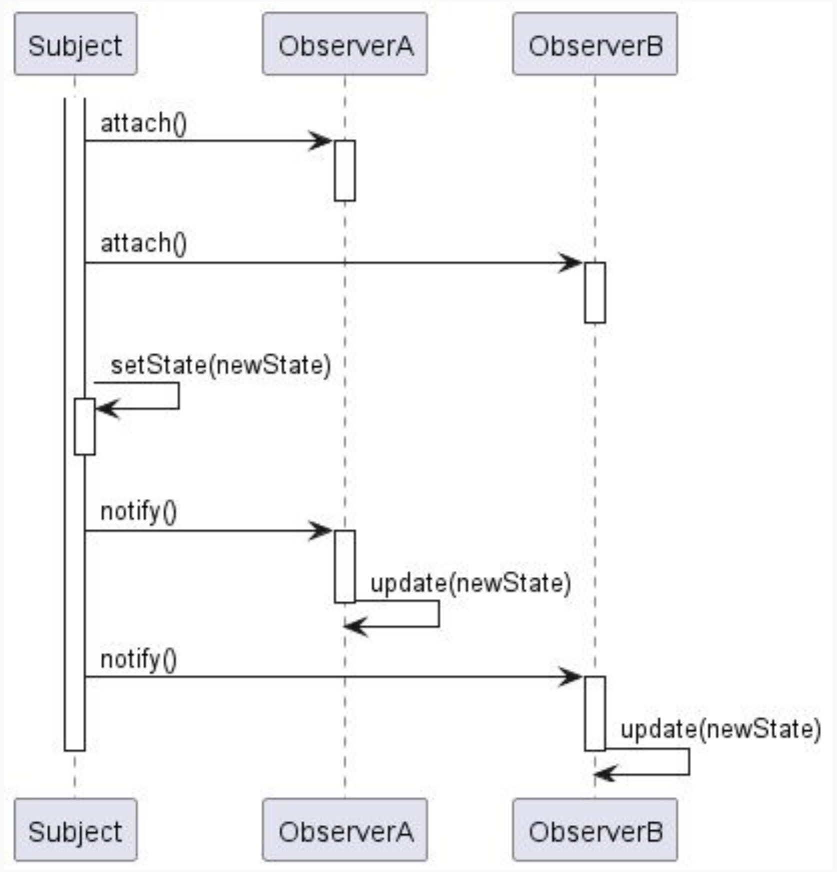


Class Diagram



C Subject
□ observers : List<Observer>
● attach(observer : Observer) : void
● detach(observer : Observer) : void
● notifyObservers() : void
► notifies
I Observer
● update() : void
C ConcreteObserverA
● update() : void
C ConcreteObserverB
● update() : void

Sequence Diagram



Code Sample

Use cases

- General
    - Event Management Systems
    - Financial Markets
    - Weather Monitoring

- In Test Automation
    - Logging Mechanism
    - Test Result Reporting

Verisoft

Happy Coding