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:** 25-03-2024

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

EXPERIMENT NO. 7

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

AIM: Implement the Observer Pattern using any language of your choice for any real-life scenario. (For example, implement a system using Observer Pattern in which registered investors are notified every time a stock changes value).

DESCRIPTION:

Observer Pattern is one of the behavioural design patterns. Observer design pattern is useful when you are interested in the state of an object and want to get notified whenever there is any change. In observer pattern, the object that watch on the state of another object are called Observer and the object that is being watched is called Subject.

SOURCE CODE:

```
class Subject:
    def __init__(self):
        self.observers = []
    def attach(self, observer):
        self.observers.append(observer)
    def detach(self, observer):
        self.observers.remove(observer)
    def notify(self):
        for observer in self.observers:
            observer.update(self)
class Stock(Subject):
    def __init__(self, symbol, price):
        super().__init__()
```

```
        self.symbol = symbol
        self.price = price
    def set_price(self, price):
        self.price = price
        self.notify()
class Investor:
    def __init__(self, name):
        self.name = name
    def update(self, subject):
        print(f"{self.name} received an update: {subject.symbol} is now {subject.price}")
# Client code
stock = Stock("AAPL", 150.0)
investor1 = Investor("John")
investor2 = Investor("Sarah")
stock.attach(investor1)
stock.attach(investor2)
stock.set_price(160.0)
stock.detach(investor2)
stock.set_price(170.0)
```

OUTPUT

```
John received an update: AAPL is now 160.0
Sarah received an update: AAPL is now 160.0
John received an update: AAPL is now 170.0
```

CONCLUSION:

We have a Subject class that represents a stock, and an Investor class that represents an investor. The Subject class has a list of observers (investors), and methods to attach, detach, and notify observers. The Stock class inherits from Subject and represents a stock with a symbol and price. The set_price method updates the price of the stock and notifies all registered investors. The Investor class has a name and an update method that receives a Subject instance and prints a message with the updated stock symbol and price. In the client code, we create a Stock instance with a symbol and price. We then create two Investor instances and attach them to the stock.

We update the price of the stock, which triggers a notification to both investors. We then detach one investor and update the stock price again, which only triggers a notification to the remaining investor.

REFERENCES:

[1] https://www.digitalocean.com/community/tutorials/observer-design-pattern-in-java