**Module 1 - Design Patterns and Principles**

**Exercise 7: Implementing the Observer Pattern**

**Aim:**

To develope a stock market monitoring application where multiple clients need to be notified whenever stock prices change using the Observer Pattern to achieve this.

**Code:**

import java.util.\*;

interface Stock {

    void register(Observer o);

    void deregister(Observer o);

    void notifyObservers();

}

class StockMarket implements Stock {

    private List<Observer> observers = new ArrayList<>();

    private double price;

    public void setPrice(double price) {

        this.price = price;

        notifyObservers();

    }

    public void register(Observer o) {

        observers.add(o);

    }

    public void deregister(Observer o) {

        observers.remove(o);

    }

    public void notifyObservers() {

        for (Observer o : observers) {

            o.update(price);

        }

    }

}

interface Observer {

    void update(double price);

}

class MobileApp implements Observer {

    public void update(double price) {

        System.out.println("MobileApp received update: Stock Price changed to " + price);

    }

}

class WebApp implements Observer {

    public void update(double price) {

        System.out.println("WebApp received update: Stock Price changed to " + price);

    }

}

public class ObserverPatternExample {

    public static void main(String[] args) {

        StockMarket market = new StockMarket();

        Observer mobile = new MobileApp();

        Observer web = new WebApp();

        market.register(mobile);

        market.register(web);

        market.setPrice(105.5);

    }

}

**Output:**

