**Implementing the Observer Pattern**

**ObserverPatternExample:**

interface Stock {

    void register(Observer obs);

    void deregister(Observer obs);

    void notifyObservers();

    void setPrice(double price);

}

interface Observer {

    void update(double price);

}

class StockMarket implements Stock {

    private Observer[] observers = new Observer[10];

    private int count = 0;

    private double stockPrice;

    public void register(Observer obs) {

        observers[count++] = obs;

    }

    public void deregister(Observer obs) {

        for (int i = 0; i < count; i++) {

            if (observers[i] == obs) {

                observers[i] = observers[--count];

                observers[count] = null;

                break;

            }

        }

    }

    public void notifyObservers() {

        for (int i = 0; i < count; i++) {

            observers[i].update(stockPrice);

        }

    }

    public void setPrice(double price) {

        System.out.println("\nStock price has been changed to ₹" + price);

        stockPrice = price;

        notifyObservers();

    }

}

class MobileApp implements Observer {

    public void update(double price) {

        System.out.println("MobileApp Price: ₹" + price);

    }

}

class WebApp implements Observer {

    public void update(double price) {

        System.out.println("WebApp Price: ₹" + 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(5000.0);

        market.setPrice(10000.0);

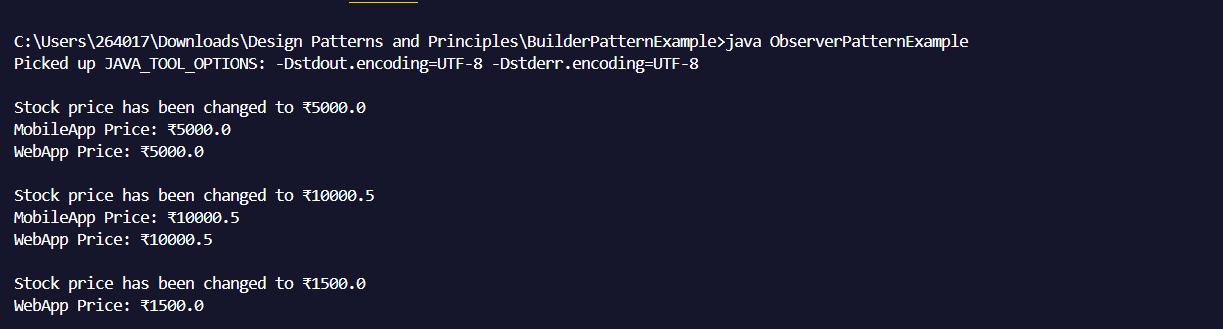
        market.deregister(mobile);

        market.setPrice(1500.0);

    }

}

**Output:**

****