Experiment – 15

Aim –Observer design pattern (Behavioral pattern)

Concept - **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.**

RestaurantOrder.java

```
import java.util.ArrayList;
import java.util.List;

public class RestaurantOrder {

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

    public int getOrder() {
        return order;
    }

    public void setOrder(int order) {
        this.order = order;
        notifyAllObservers();
    }

    public void attach(Observer observer){
        observers.add(observer);
    }

    public void notifyAllObservers(){
        for (Observer observer : observers) {
            observer.update();
        }
    }
}
```

Observer.java

```
public abstract class Observer {
    protected RestaurantOrder restaurantOrder;
    public abstract void update();
}
```

OrderConfirmed.java

```

public class OrderConfirmed extends Observer{
    public OrderConfirmed(RestaurantOrder restaurantOrder) {
        this.restaurantOrder = restaurantOrder;
        this.restaurantOrder.attach(this);
    }

```

```java
    @Override
    public void update() {
        System.out.println("Order confirm");
    }
}
```

OrderDelivered.java

```java
public class OrderDelivered extends Observer{
    public OrderDelivered(RestaurantOrder restaurantOrder) {
        this.restaurantOrder = restaurantOrder;
        this.restaurantOrder.attach(this);
    }

    @Override
    public void update() {
        System.out.println("Order deliver");
    }
}
```

OrderCancelled.java

```java
public class OrderCancelled extends Observer{
    public OrderCancelled(RestaurantOrder restaurantOrder) {
        this.restaurantOrder = restaurantOrder;
        this.restaurantOrder.attach(this);
    }

    @Override
    public void update() {
        System.out.println("Order Cancel");
    }
}
```

Client.java

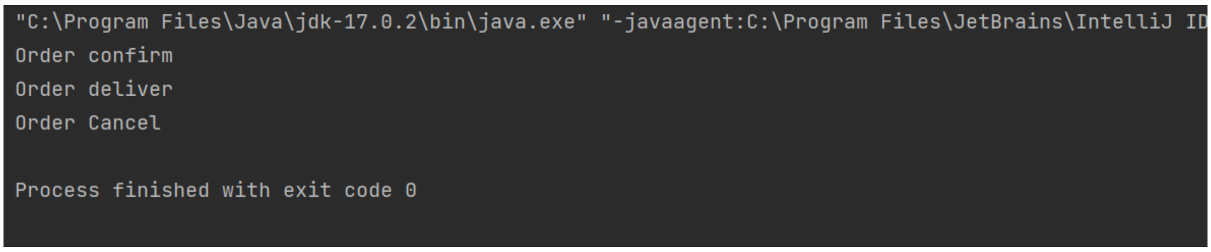
```java
public class Client {
    public static void main(String[] args) {
        RestaurantOrder restaurantOrder = new RestaurantOrder();

        new OrderConfirmed(restaurantOrder);
        new OrderDelivered(restaurantOrder);
        new OrderCancelled(restaurantOrder);


        restaurantOrder.setOrder(1);

    }
}
```

OUTPUT :



```
"C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ ID
Order confirm
Order deliver
Order Cancel

Process finished with exit code 0
```