Observer Design Pattern.

1. What is **Observer Design Pattern?**

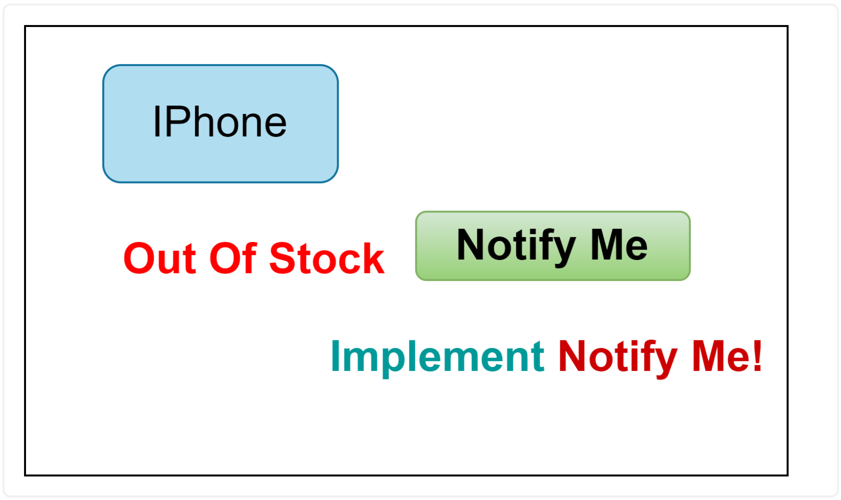
Amazon Interview Question On Observer Design Pattern.

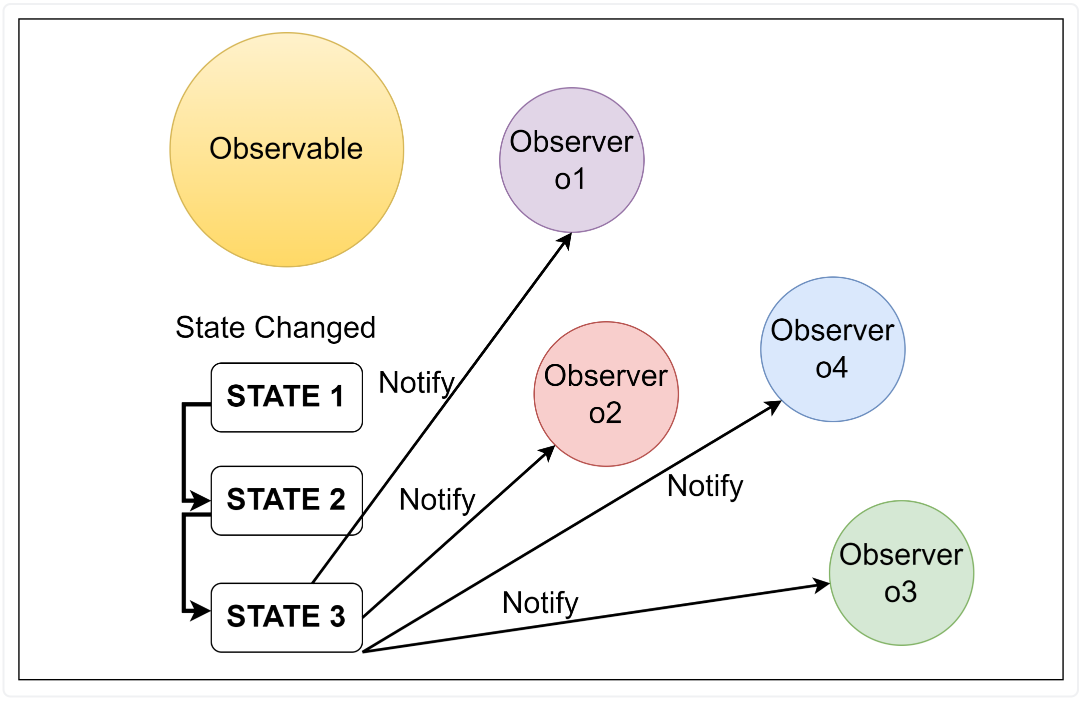
- After product comes into stock we have to send notification to each and every customer who done click notify Me button for this product.
- We have **Observable Interface.**
- **Observable Interface**

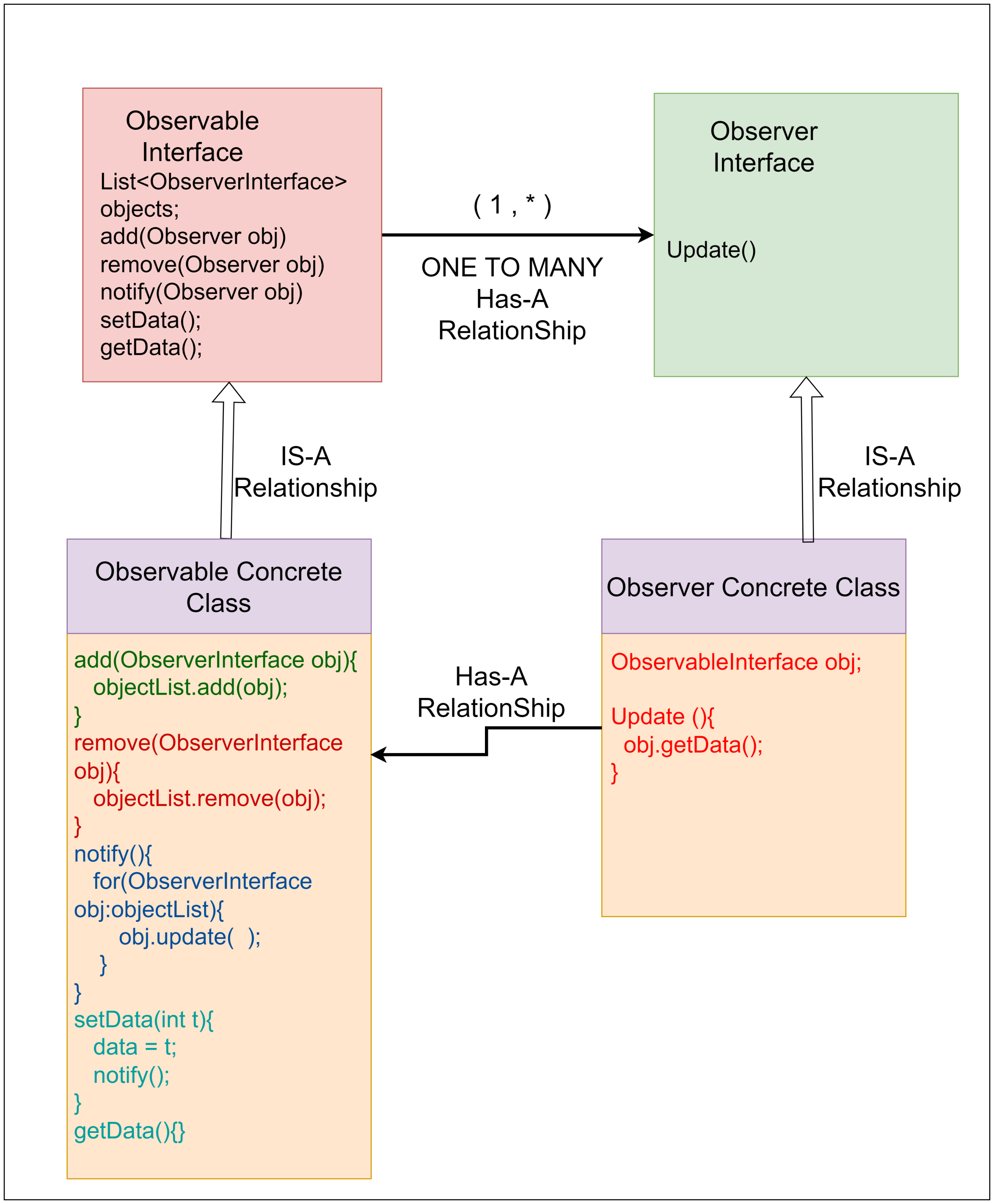
add(ObserverInterface obj) or registration
remove(ObserverInterface obj)
notify()

- **Observer Interface**

 update()




Observable
Observer o1
State Changed
STATE 1
Notify
Observer o2
Observer o4
STATE 2
Notify
Notify
STATE 3
Notify
Observer o3



SOLUTION FOR Amazon Interview Questions.

```
package
System_Design.DesignPatterns.ObserverDesignPattern.Amazon_Interview_Question.Observable;

import
System_Design.DesignPatterns.ObserverDesignPattern.Amazon_Interview_Question.Observer.No
tificationAlertObserver;

public interface StockObservable {
        public void add(NotificationAlertObserver obj);
        public void remove(NotificationAlertObserver obj);
        public void setStockCount(int newStockCount);
        public void notifySubscribers();
        public int getStockCount();
}
```

```
package
System_Design.DesignPatterns.ObserverDesignPattern.Amazon_Interview_Question.Observable;

import
System_Design.DesignPatterns.ObserverDesignPattern.Amazon_Interview_Question.Observer.No
tificationAlertObserver;

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

public class IphoneObservableImple implements StockObservable{

    public List<NotificationAlertObserver> observerList = new ArrayList<>();
    public int stockCount = 0;

    @Override
    public void add(NotificationAlertObserver obj) {
        observerList.add(obj);
    }

    @Override
    public void remove(NotificationAlertObserver obj) {

        observerList.remove(obj);
    }

    @Override
    public void setStockCount(int newStockCount) {
        if (stockCount == 0){
            notifySubscribers();
        }
        stockCount = stockCount+newStockCount;
    }

    @Override
    public void notifySubscribers() {
        for (NotificationAlertObserver observer : observerList){
            observer.update();
        }

    }
    @Override
    public int getStockCount() {
        return stockCount;
    }
}
```

```
package
System_Design.DesignPatterns.ObserverDesignPattern.Amazon_Interview_Question.Observer;

public interface NotificationAlertObserver {
    public void update();
}
```

```
package
System_Design.DesignPatterns.ObserverDesignPattern.Amazon_Interview_Question.Observer;

import
System_Design.DesignPatterns.ObserverDesignPattern.Amazon_Interview_Question.Observable.
StockObservable;

public class EmailObserverImple implements NotificationAlertObserver{

    String emailId;
    StockObservable observable;
    //Constructor Injections
    public EmailObserverImple(String emailId, StockObservable observable){
        this.emailId=emailId;
        this.observable = observable;
    }
    @Override
    public void update() {
        sendEmail(emailId,"Product Is In-Stock hurry up!");

    }

    private void sendEmail(String emailId, String s) {
        System.out.println("mail sent to: "+ emailId);
    }
}
```

```
package
System_Design.DesignPatterns.ObserverDesignPattern.Amazon_Interview_Question.Observer;

import
System_Design.DesignPatterns.ObserverDesignPattern.Amazon_Interview_Question.Observable.
```

```
StockObservable;

public class MobileObserverImple implements NotificationAlertObserver{
    String phone;
    StockObservable observable;
    public MobileObserverImple(String phone,StockObservable observable){
        this.observable=observable;
        this.phone=phone;
    }



    @Override
    public void update() {
        sendMsg(phone,"Product is in stock hurry up.");
        //send the actual email to
    }

    private void sendMsg(String phone, String s) {
        System.out.println("message sent to phone: " +phone);
    }
}
```

```
package System_Design.DesignPatterns.ObserverDesignPattern.Amazon_Interview_Question;

import
System_Design.DesignPatterns.ObserverDesignPattern.Amazon_Interview_Question.Observable.
IphoneObservableImple;
import
System_Design.DesignPatterns.ObserverDesignPattern.Amazon_Interview_Question.Observable.
StockObservable;
import
System_Design.DesignPatterns.ObserverDesignPattern.Amazon_Interview_Question.Observer.Em
ailObserverImple;
import
System_Design.DesignPatterns.ObserverDesignPattern.Amazon_Interview_Question.Observer.Mo
bileObserverImple;
import
System_Design.DesignPatterns.ObserverDesignPattern.Amazon_Interview_Question.Observer.No
tificationAlertObserver;

public class Store {
    public static void main(String[] args){
        StockObservable iphoneObservable = new IphoneObservableImple();
        NotificationAlertObserver observer1 = new
EmailObserverImple("sauravsrivastava121@gmail.com",iphoneObservable);
        NotificationAlertObserver observer2 = new
EmailObserverImple("gauravsrivastava2003mth@gmail.com",iphoneObservable);
        NotificationAlertObserver observer3 = new
MobileObserverImple("7061036502",iphoneObservable);

        iphoneObservable.add(observer1);
        iphoneObservable.add(observer2);
        iphoneObservable.add(observer3);

        iphoneObservable.setStockCount(10);
    }
}
```