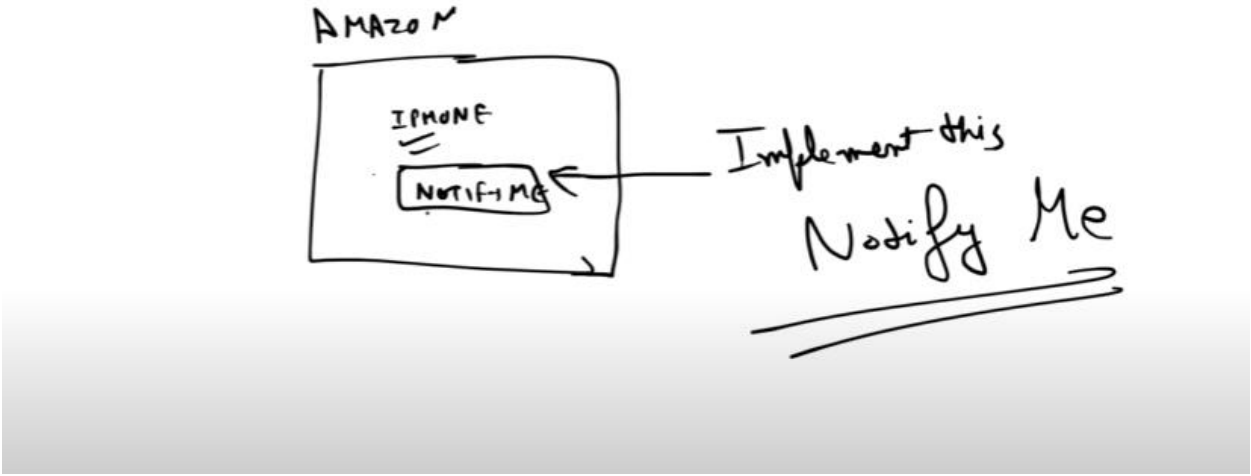
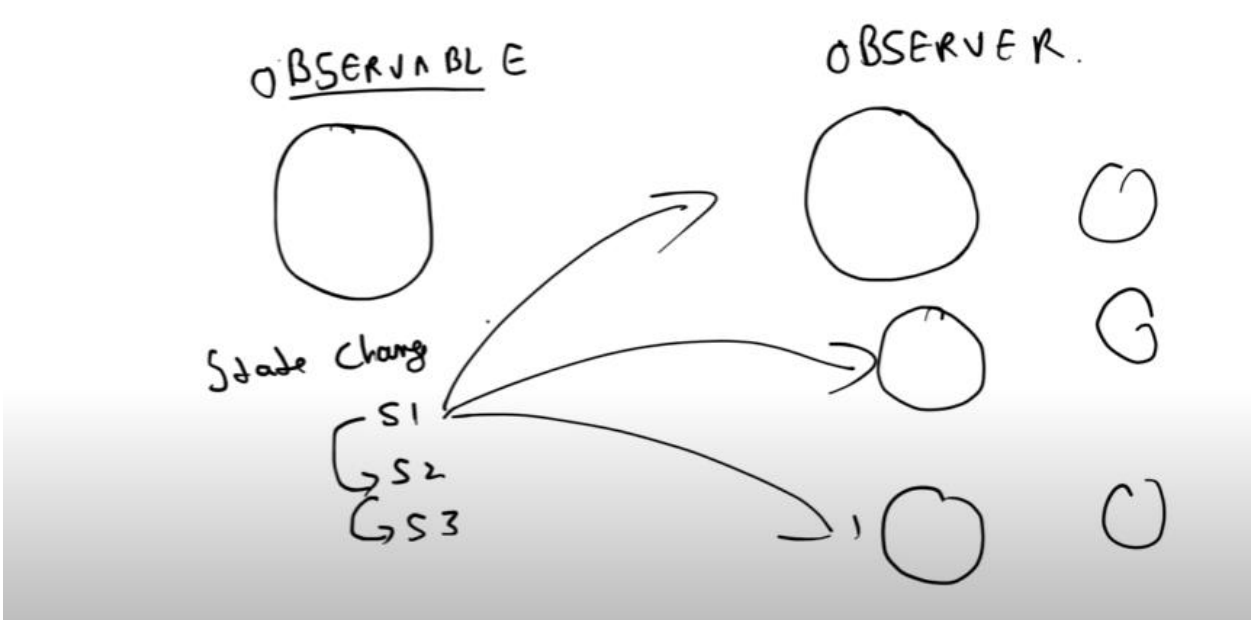
Observer Design Pattern - Walmart Interview Question

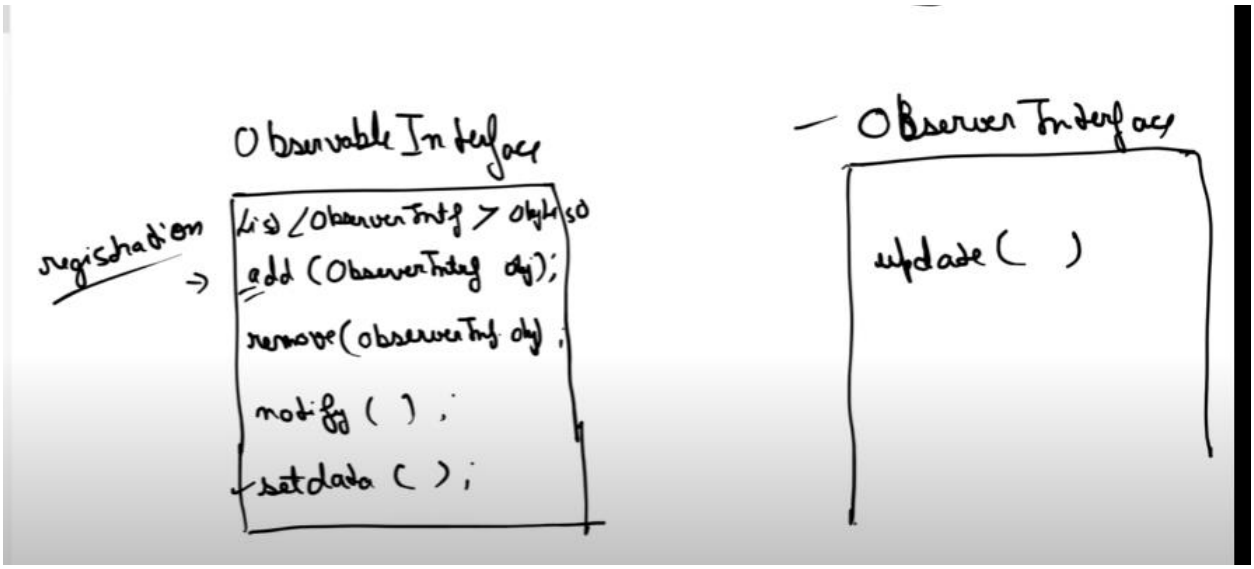
Tuesday, 10 May 2022 3:44 PM



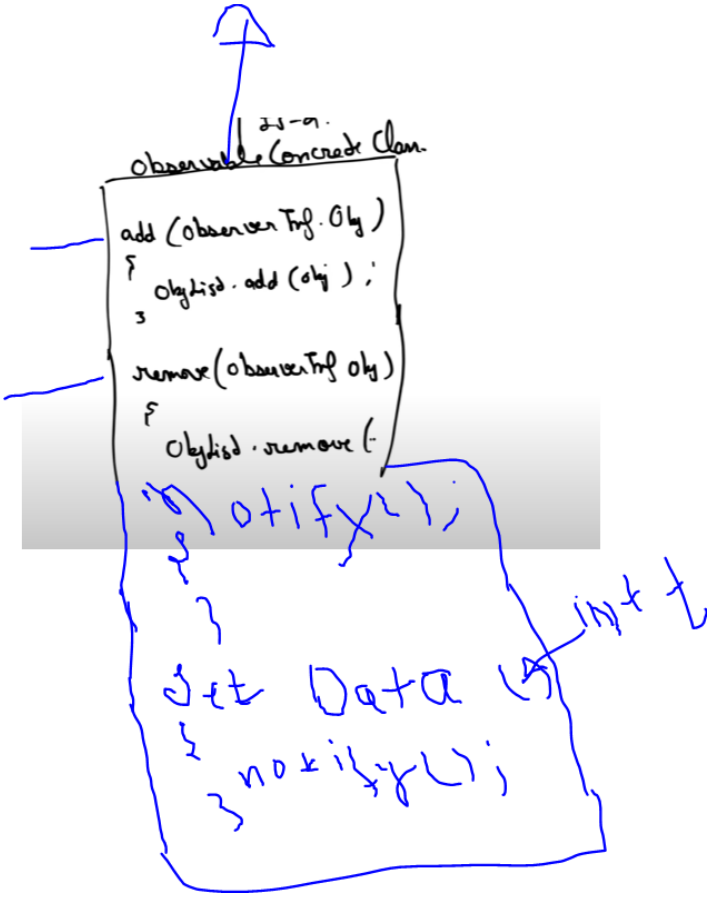
Question : Send notification to all observer who has opted for Notify me

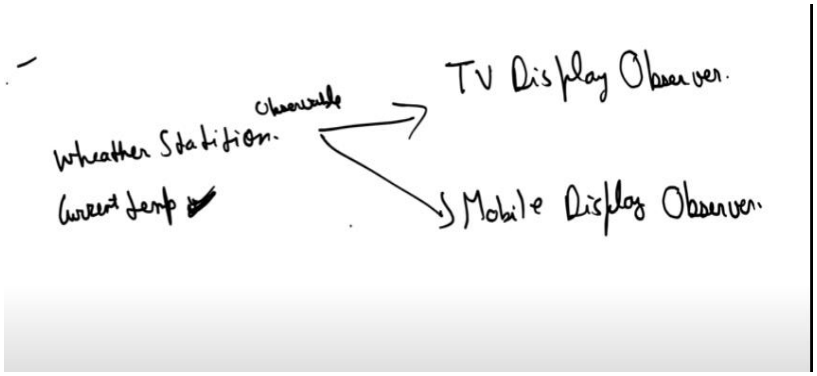


When state changes of Observable from s1 to s2 .. , send update to all observers

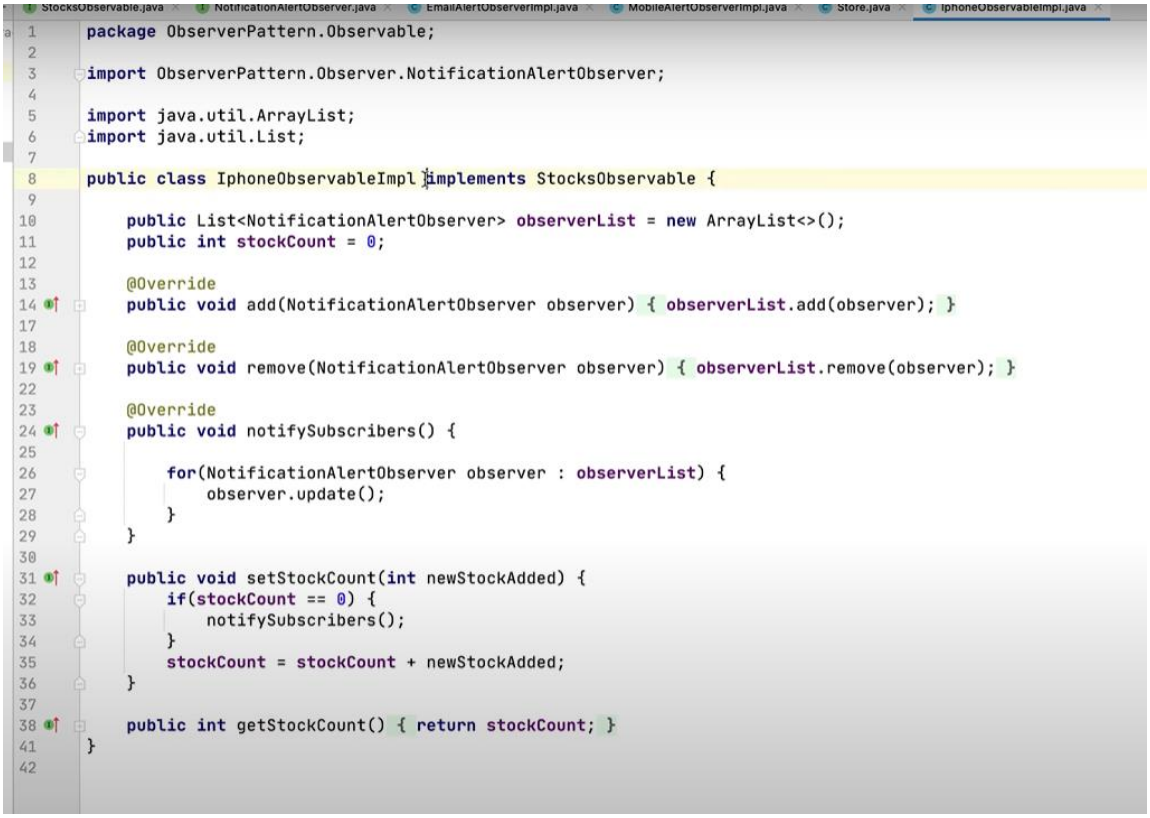


Has a relationship (0,*) ---> one to many , many observer can observe the observable .
Make a concrete class of observable Interface.



e.g. 


```java
package ObserverPattern.Observable;

import ObserverPattern.Observer.NotificationAlertObserver;

public interface StockObservable {

    public void add(NotificationAlertObserver observer);

    public void remove(NotificationAlertObserver observer);

    public void notifySubscribers();

    public void setStockCount(int newStockAdded);

    public int getStockCount();
}
```



```java
package ObserverPattern.Observable;

import ObserverPattern.Observer.NotificationAlertObserver;

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

public class IphoneObservableImpl implements StocksObservable {

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

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

    @Override
    public void remove(NotificationAlertObserver observer) { observerList.remove(observer); }

    @Override
    public void notifySubscribers() {

        for(NotificationAlertObserver observer : observerList) {
            observer.update();
        }
    }

    public void setStockCount(int newStockAdded) {
        if(stockCount == 0) {
            notifySubscribers();
        }
        stockCount = stockCount + newStockAdded;
    }

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

```java
package ObserverDesignPattern.Observable;

import
ObserverDesignPattern.Observer.NotificationAlertObs
erver;

public interface StockObservable {
    void add(NotificationAlertObserver observer);


    void remove(NotificationAlertObserver
observer);
    void notifySubscriber();
    void setStockCount(int newStockAdded);
    int getStockCount();

}
```

```java
package ObserverDesignPattern.Observer;


// Observe Product stock , is it available or not ,
// yes then alert
public interface NotificationAlertObserver {
    public abstract void update();
}
```

```java
package ObserverDesignPattern.Observable;

import
ObserverDesignPattern.Observer.NotificationAlertObs
erver;

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

public class IphoneOObservableImpl implements
StockObservable{
   // all the customer opted for notify me for
iphone , customer can be of type email or mobile
    public List<NotificationAlertObserver>
observerList = new ArrayList<>();
    public int stockCount =0;
    @Override
    public void add(NotificationAlertObserver
observer) {
        observerList.add(observer); // add all
observer or customer
    }

    @Override
    public void remove(NotificationAlertObserver
observer) {
        this.observerList.remove(observer); //
remove observer
```

```
    }

    @Override
    public void notifySubscriber() {
        // notify each customer i.e. observer
        for(NotificationAlertObserver observer :
observerList)
        {
            observer.update();
        }
    }

    @Override
    public void setStockCount( int newStockAdded )
{
        // if this method is being called ,
        // means state is changing ,so notify
        // if stockCount was 0 ,means new stock are
coming in here
        // Product are coming out of stock ,means
product is available
        if(stockCount ==0)
        {
            notifySubscriber();
        }
        stockCount =  stockCount + newStockAdded;
    }

    @Override
    public int getStockCount() {

        return stockCount;
    }
}
package ObserverDesignPattern.Observer;

import
ObserverDesignPattern.Observable.StockObservable;
```

```
public class EmailAlertObserverimpl implements
NotificationAlertObserver{
    String emailId;
    StockObservable observable;
    public EmailAlertObserverimpl(String emaildId ,
StockObservable observable) {
        this.emailId = emaildId;
        this.observable = observable;

    }

    @Override
    public void update() {
    senMail(emailId,"product is in stock");
    }

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

```
package ObserverDesignPattern.Observer;

import
ObserverDesignPattern.Observable.StockObservable;

public class MobileAlertObserverImpl implements
NotificationAlertObserver{
String userName;
StockObservable observable;

    public MobileAlertObserverImpl(String userName,
StockObservable observable) {
```

```
        this.userName = userName;
        this.observable  = observable;
    }

    @Override
    public void update() {
        sendMsgOnMobile(userName,"product is in
stock");
    }
    private void sendMsgOnMobile(String userName,
String msg)
    {
        System.out.println("msg sent to" + userName
+ msg);
    }
}
```

```
package ObserverDesignPattern;

import
ObserverDesignPattern.Observable.IphoneOObservableI
mpl;
import
ObserverDesignPattern.Observable.StockObservable;
import
ObserverDesignPattern.Observer.EmailAlertObserverim
pl;
import
ObserverDesignPattern.Observer.MobileAlertObserverI
mpl;
import
ObserverDesignPattern.Observer.NotificationAlertObs
erver;

public class Stock {
    public static void main(String[] args) {
        // customer comes and click on notify me
```

```
for iphone product
        StockObservable iphoneStockObeservable =
new IphoneOObservableImpl();
        // choose to notify via email
        NotificationAlertObserver observer1 = new
EmailAlertObserverimpl("abc@gmail.com",iphoneStockO
beservable);
        NotificationAlertObserver observer2 = new
MobileAlertObserverImpl("nitish",iphoneStockObeserv
able);
        iphoneStockObeservable.add(observer1);
        iphoneStockObeservable.add(observer2);
        iphoneStockObeservable.setStockCount(10);
    }
}
```