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. In observer pattern, the object that watch on the state of another object are called **Observer** and the object that is being watched is called **Subject** or **Observable** .

**Subject** contains a list of observers to notify of any change in it’s state, so it should provide methods using which observers can register and unregister themselves. Subject also contain a method to notify all the observers of any change.

Observer should have a method to set the object to watch and another method that will be used by Subject to notify them of any updates.

It is also known as **publish and subscribe** model , we could implement a simple topic and observers can register to this topic. Whenever any new message will be posted to the topic, all the registers observers will be notified and they can consume the message.

It is a behavioral design pattern.

UseCase : Notify users when the stock of an Item is available through different channels like mobile number / email .

public interface StocksObservable{  
 public void add(NotificationAlertObserver notificationAlertObserver) ;  
  
 public void remove(NotificationAlertObserver notificationAlertObserver) ;  
  
 public void notifyAllSubscribers() ;  
  
 public void updateStockCount(int stock) ;  
  
 public int getStockCount() ;  
}

public class IphoneObservableImpl implements StocksObservable {  
  
 List<NotificationAlertObserver> observerList = new ArrayList<>();  
 int stockCount = 0 ;  
  
 @Override  
 public void add(NotificationAlertObserver notificationAlertObserver) {  
 observerList.add(notificationAlertObserver) ;  
 }  
  
 @Override  
 public void remove(NotificationAlertObserver notificationAlertObserver) {  
 observerList.remove(notificationAlertObserver) ;  
 }  
  
 @Override  
 public void notifyAllSubscribers() {  
 for(NotificationAlertObserver observer : observerList ){  
 observer.update();  
 }  
 }  
  
 @Override  
 public void updateStockCount(int newStock) {  
  
 if(stockCount + newStock >= 0 ) {  
 stockCount += newStock ;  
  
 if(stockCount > 0 ){  
 notifyAllSubscribers();  
 }  
 }  
  
 }  
  
 @Override  
 public int getStockCount() {  
 return stockCount ;  
 }  
}

public class EmailAlertObserverImpl implements NotificationAlertObserver {  
  
 String userName ;  
 String emailId ;  
 StocksObservable observable ;  
  
 public EmailAlertObserverImpl(String userName , String emailId , StocksObservable observable ){  
 this.userName = userName ;  
 this.emailId = emailId ;  
 this.observable = observable ;  
 }  
  
 @Override  
 public void update() {  
 sendEmail( userName , emailId , " HurryUp ! Product is available now in the stock with quantity : " +  
 observable.getStockCount() );  
 }  
  
 private void sendEmail( String userName , String emailId , String message ){  
 System.*out*.println("Message is sent to " + userName + " and the email " + emailId + message );  
 }  
}

public class MobileAlertObserverimpl implements NotificationAlertObserver {  
  
 String userName ;  
 String mobileNumber ;  
 StocksObservable observable ;  
  
 public MobileAlertObserverimpl( String userName , String mobileNumber , StocksObservable observable ){  
 this.userName = userName ;  
 this.mobileNumber = mobileNumber ;  
 this.observable = observable ;  
 }  
  
 @Override  
 public void update() {  
 sendEmail( userName , mobileNumber , " HurryUp ! Product is available now in the stock with quantity : " +  
 observable.getStockCount() );  
 }  
  
 private void sendEmail(String userName, String mobileNumber, String message ) {  
 System.*out*.println("Message is sent to " + userName + " and the mobile " + mobileNumber + message );  
 }  
  
  
}

public class ObserverDesignPattern {  
  
 public static void main(String[] args) {  
  
 StocksObservable iphoneStocksObservable = new IphoneObservableImpl() ;  
  
 NotificationAlertObserver observer1 = new EmailAlertObserverImpl("Sesh" , "pari@gmail.com" , iphoneStocksObservable ) ;  
  
 NotificationAlertObserver observer2 = new EmailAlertObserverImpl("Sesh" , "sesh@gmail.com" , iphoneStocksObservable ) ;  
  
 NotificationAlertObserver observer3 = new MobileAlertObserverimpl("Sesh" , "1234567890" , iphoneStocksObservable ) ;  
  
 iphoneStocksObservable.add(observer1);  
 iphoneStocksObservable.add(observer2);  
 iphoneStocksObservable.add(observer3);  
  
 iphoneStocksObservable.updateStockCount(10);  
  
 iphoneStocksObservable.updateStockCount(-10);  
  
 iphoneStocksObservable.updateStockCount(1);  
  
 iphoneStocksObservable.updateStockCount(-10);  
  
 }  
}

Output :

Message is sent to Sesh and the email pari@gmail.com HurryUp ! Product is available now in the stock with quantity : 10

Message is sent to Sesh and the email sesh@gmail.com HurryUp ! Product is available now in the stock with quantity : 10

Message is sent to Sesh and the mobile 1234567890 HurryUp ! Product is available now in the stock with quantity : 10

Message is sent to Sesh and the email pari@gmail.com HurryUp ! Product is available now in the stock with quantity : 1

Message is sent to Sesh and the email sesh@gmail.com HurryUp ! Product is available now in the stock with quantity : 1

Message is sent to Sesh and the mobile 1234567890 HurryUp ! Product is available now in the stock with quantity : 1