https://www.baeldung.com/java-observer-pattern

Publisher/Subscriber without Observer Pattern

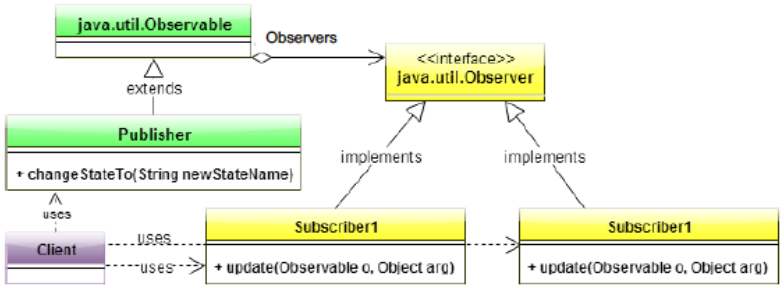
```java
import java.util.ArrayList;
import java.util.List;

public class NewsAgency {
    private String news;
    private List<Channel> channels = new ArrayList<>();

    public void addObserver(Channel channel) {
        this.channels.add(channel);
    }

    public void removeObserver(Channel channel) {
        this.channels.remove(channel);
    }

    public void setNews(String news) {
        this.news = news;
        for (Channel channel : this.channels) {
            channel.update(this.news);
        }
    }
}
```

```java
public interface Channel {
    public void update(Object o);
}
```

```java
public class NewsChannel implements Channel {

    private String news;

    @Override
    public void update(Object news) {
        this.setNews((String) news);
    }

    public String getNews() {
        return news;
    }

    public void setNews(String news) {
        this.news = news;
    }

}
```

Publisher/Subscriber with Observer Pattern

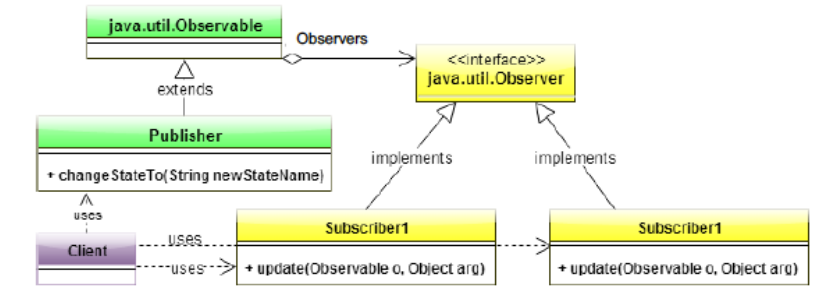
```java
import java.util.Observable;

public class ONewsAgency extends Observable {

    public void setNews(String news) {
        this.setChanged();
        this.notifyObservers(news);
    }
}
```




```java
import java.util.Observable;
import java.util.Observer;

public class ONewsChannel implements Observer {

    private String news;

    @Override
    public void update(Observable o, Object news) {
        this.setNews((String) news);
    }

    public String getNews() {
        return news;
    }

    public void setNews(String news) {
        this.news = news;
    }
}
```

Publisher/Subscriber with Observer Pattern

```java
import java.util.Observable;

public class ONewsAgency extends Observable {

    public void setNews(String news) {
        this.setChanged();
        this.notifyObservers(news);
    }
}
```




```java
import java.util.Observable;
import java.util.Observer;

public class ONewsChannel implements Observer {

    private String news;

    @Override
    public void update(Observable o, Object news) {
        this.setNews((String) news);
    }

    public String getNews() {
        return news;
    }

    public void setNews(String news) {
        this.news = news;
    }
}
```

Publisher/Subscriber with *PropertyChangeListener*

```java
import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeSupport;

public class PCLNewsAgency {

    private String news;
    private PropertyChangeSupport support;

    public PCLNewsAgency() {
        support = new PropertyChangeSupport(this);
    }

    public void addPropertyChangeListener(PropertyChangeListener pcl) {
        support.addPropertyChangeListener(pcl);
    }

    public void removePropertyChangeListener(PropertyChangeListener pcl) {
        support.removePropertyChangeListener(pcl);
    }

    public void setNews(String value) {
        support.firePropertyChange("news", this.news, value);
        this.news = value;

    }
}
```

```java
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

public class PCLNewsChannel implements PropertyChangeListener {

    private String news;

    public void propertyChange(PropertyChangeEvent evt) {
        this.setNews((String) evt.getNewValue());
    }

    public String getNews() {
        return news;
    }

    public void setNews(String news) {
        this.news = news;
    }
}
```

```java
import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class ObserverIntegrationTest {

    @Test
    public void whenChangingNewsAgencyState_thenNewsChannelNotified() {

        NewsAgency observable = new NewsAgency();
        NewsChannel observer1 = new NewsChannel();
        NewsChannel observer2 = new NewsChannel();

        observable.addObserver(observer1);
        observable.addObserver(observer2);

        observable.setNews("newsA");
        System.out.println("New state received by NewsChannel1: "+observer1.getNews());
        assertEquals(observer1.getNews(), "newsA");
        System.out.println("New state received by NewsChannel2: "+observer2.getNews());
        assertEquals(observer2.getNews(), "newsA");

        observable.setNews("newsB");
        System.out.println("New state received by NewsChannel1: "+observer1.getNews());
        assertEquals(observer1.getNews(), "newsB");
        System.out.println("New state received by NewsChannel2: "+observer2.getNews());
        assertEquals(observer2.getNews(), "newsB");
    }
```

Output:

```
New state received by PCLNewsChannel1: PCLnewsA
New state received by PCLNewsChannel2: PCLnewsA
New state received by PCLNewsChannel1: PCLnewsB
New state received by PCLNewsChannel2: PCLnewsB

New state received by ONewsChannel1: OnewsA
New state received by ONewsChannel2: OnewsA
New state received by ONewsChannel1: OnewsB
New state received by ONewsChannel2: OnewsB

New state received by NewsChannel1: newsA
New state received by NewsChannel2: newsA
New state received by NewsChannel1: newsB
New state received by NewsChannel2: newsB
```

```java
    @Test
    public void whenChangingONewsAgencyState_thenONewsChannelNotified() {

        ONewsAgency observable = new ONewsAgency();
        ONewsChannel observer1 = new ONewsChannel();
        ONewsChannel observer2 = new ONewsChannel();

        observable.addObserver(observer1);
        observable.addObserver(observer2);

        observable.setNews("OnewsA");
        System.out.println("New state received by ONewsChannel1: "+observer1.getNews());
        assertEquals(observer1.getNews(), "OnewsA");
        System.out.println("New state received by ONewsChannel2: "+observer2.getNews());
        assertEquals(observer2.getNews(), "OnewsA");

        observable.setNews("OnewsB");
        System.out.println("New state received by ONewsChannel1: "+observer1.getNews());
        assertEquals(observer1.getNews(), "OnewsB");
        System.out.println("New state received by ONewsChannel2: "+observer2.getNews());
        assertEquals(observer2.getNews(), "OnewsB");
    }


    @Test
    public void whenChangingPCLNewsAgencyState_thenPCLNewsChannelNotified() {

        PCLNewsAgency observable = new PCLNewsAgency();
        PCLNewsChannel observer1 = new PCLNewsChannel();
        PCLNewsChannel observer2 = new PCLNewsChannel();

        observable.addPropertyChangeListener(observer1);
        observable.addPropertyChangeListener(observer2);

        observable.setNews("PCLnewsA");
        System.out.println("New state received by PCLNewsChannel1: "+observer1.getNews());
        assertEquals(observer1.getNews(), "PCLnewsA");
        System.out.println("New state received by PCLNewsChannel2: "+observer2.getNews());
        assertEquals(observer2.getNews(), "PCLnewsA");

        observable.setNews("PCLnewsB");
        System.out.println("New state received by PCLNewsChannel1: "+observer1.getNews());
        assertEquals(observer1.getNews(), "PCLnewsB");
        System.out.println("New state received by PCLNewsChannel2: "+observer2.getNews());
        assertEquals(observer2.getNews(), "PCLnewsB");
    }
}
```