Het Observer Pattern

Observer Pattern
Doel

In het Observer Pattern kunnen objecten zich **abonneren** op berichten van een ander object.

Rollen in het pattern:

- **subject**: het object dat berichten stuurt (de ‘uitgever’)

- **observer**: een object dat de berichten van het subject ontvangt (de ‘abonnee’)

Observers kunnen zich **op elk moment** abonneren, of hun abonnement weer opzeggen.

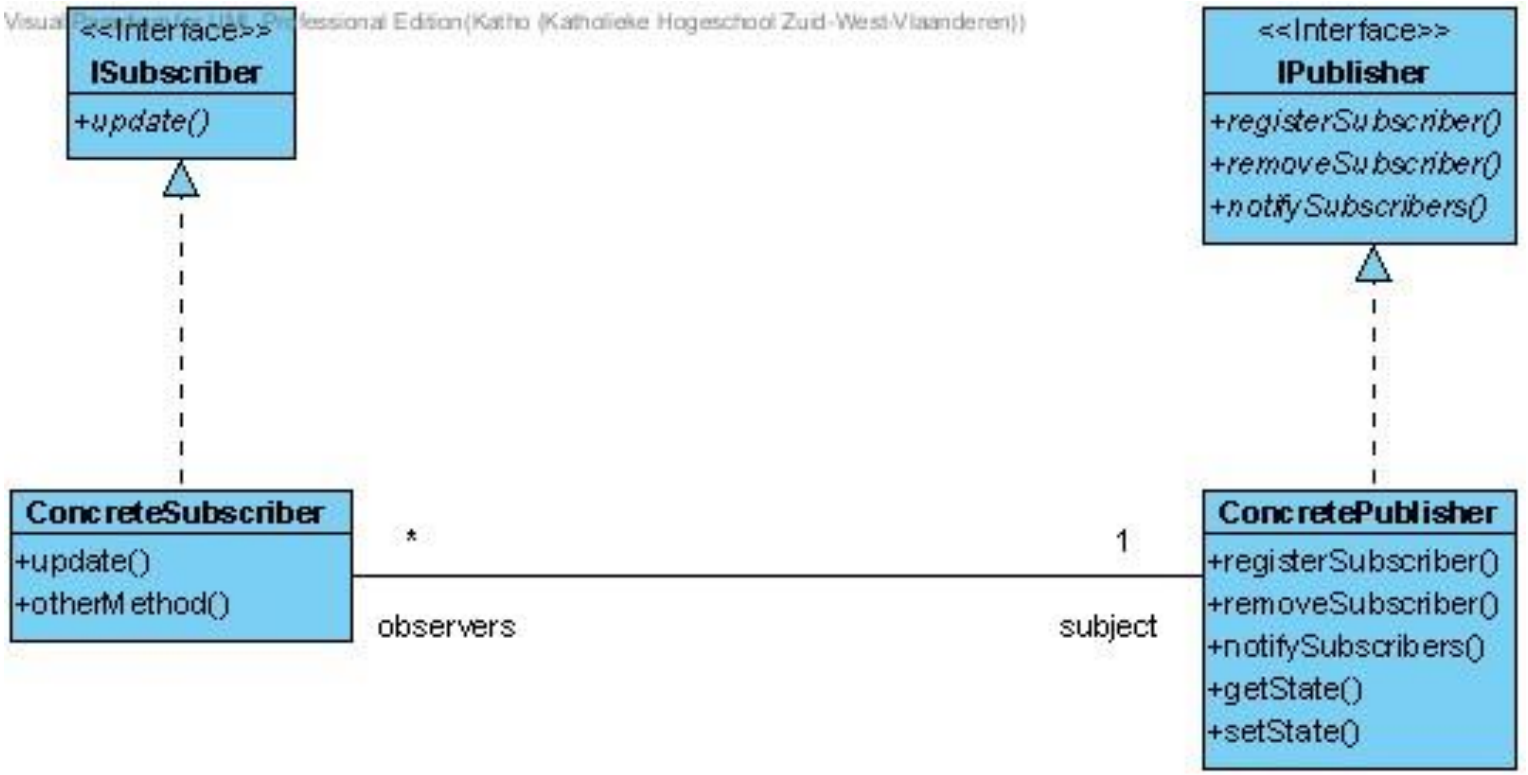
Observer Pattern
Wat

Een **één-op-veel-relatie** tussen objecten,

zodanig dat wanneer de toestand van een object verandert,

alle afhankelijke objecten **automatisch** daarvan een bericht krijgen.

Observer Pattern
Structuur



Observer Pattern
Lage Koppeling

Dit pattern is een toepassing van het **principe van lage koppeling**.

Het object dat berichten verstuurt is losgekoppeld van de ontvangers.

Dit is het tegendeel van wat je hebt bij een gewone methodeaanroep. Bij een gewone methodeaanroep is de verzender van een bericht (het object dat een methode van een ander object aanroept) heel sterk gekoppeld aan de ontvanger.

Observer Pattern
Lage Koppeling (2)

De ontkoppeling van zender en ontvanger is er door het gebruik van *interfaces*.

Een subject stuurt zijn updates naar ISubscribers, niet naar ConcreteSubscribers.

Het subject is dus niet gekoppeld aan ConcreteSubscriber.

Observer Pattern
Ontwerpprincipes

Isoleer het veranderlijke van datgene wat gelijk blijft

- Met Observer kun je het aantal observers aanpassen zonder het subject te moeten veranderen.
- Het subject kan zijn toestand veranderen zonder zijn observers te kennen.

Programmeer naar interfaces, niet naar implementaties

- Zowel de publisher als de subscriber gebruiken een interface.

Gebruik liever compositie dan overerving

- De observer gebruikt een subject, maar is niet afgeleid van een subject.

Observer Pattern in Java

Er zijn een paar alternatieven:

- Implementeer alles zelf, zonder hulp van jdk-klassen.
- Gebruik PropertyChangeListener uit java.beans
- Gebruik Flow (reactive streams - Java 9)

- De jdk-klassen Observer en Observable mag je niet meer gebruiken. Die klassen zijn deprecated.

Observer Pattern in Java
PropertyChangeListener

Publisher:

- Geen interface nodig
- Moet een veld hebben van type PropertyChangeSupport

Subscriber:

- Moet de interface PropertyChangeListener implementeren
- Die interface heeft een methode propertyChange(), met als argument een PropertyChangeEvent.
- PropertyChangeEvent heeft getters voor de gegevens van de boodschap.

Al die klassen zitten in java.beans .

Observer Pattern in Java
Voorbeeld van een publisher


```
public class Publisher {
    private String news;
    private PropertyChangeSupport support;

    public Publisher() { support = new PropertyChangeSupport(this); }
    // subscriber aanmelden
    public void addPropertyChangeListener(PropertyChangeListener pcl) {
         support.addPropertyChangeListener(pcl);
    }
    // subscriber afmelden
    public void removePropertyChangeListener(PropertyChangeListener pcl) {
         support.removePropertyChangeListener(pcl);
    }
    // boodschap versturen
    public void setNews(String value) {
        support.firePropertyChange("news", this.news, value);
        this.news = value;
    }
}
```

Een naam voor de boodschap

Oude waarde

Nieuwe waarde

Observer Pattern in Java
Voorbeeld van een subscriber


```
public class Subscriber
    implements PropertyChangeListener {
    // boodschap ontvangen
    public void propertyChange(PropertyChangeEvent evt) {
        String news = (String) evt.getNewValue();
    }

    // rest van de klasse
}
```

```
// De samenwerking van publisher en subscriber
Publisher observable = new Publisher();
Subscriber observer = new Subscriber();

observable.addPropertyChangeListener(observer);
observable.setNews("news");
```

Observer Pattern in de JDK
Observable/Observer (deprecated)

Subject-kant	Observer-kant
class java.util.Observable	interface java.util.Observer
methodes: addObserver() deleteObserver() notifyObservers() setChanged()	*methodes:* update(Observable o, Object arg)

Observer Pattern in de JDK
Obervable/Observer (deprecated)

- Een subject-klasse moet afgeleid zijn van de klasse Observable
- Observable is al functioneel, geen extra code nodig
- Observable stuurt alleen een boodschap naar de observers als vooraf setChanged() aangeroepen wordt

- update() in Observer-interface heeft twee parameters:
  - de Observable die het bericht stuurt
  - een Object dat gebruikt kan worden om data door te sturen

- Observable en Observer zijn DEPRECATED: gebruik die niet meer.

Observer Pattern in .NET

Meer info: ‘Observer Design Pattern’ op msdn (http://msdn.microsoft.com/en-us/library/ee850490.aspx)

Vanaf .NET v4 .

.NET gebruikt een **push**-model.

Observer Pattern in .NET

generiek

Subject: interface System.IObservable<T>

Observer: interface System.IObserver<T>

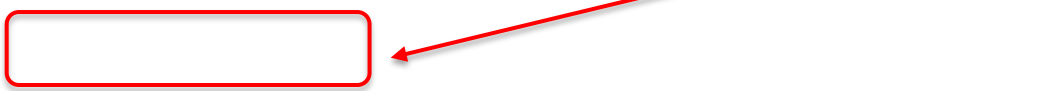
Microsoft® .NET

katholieke hogeschool associatie KU Leuven vives

Observer Pattern in .NET

IObservable<T>

De methode om een Observer toe te voegen.



Observer Pattern in .NET

Subscribe()

Subscribe() moet zelf de lijst van observers onderhouden.

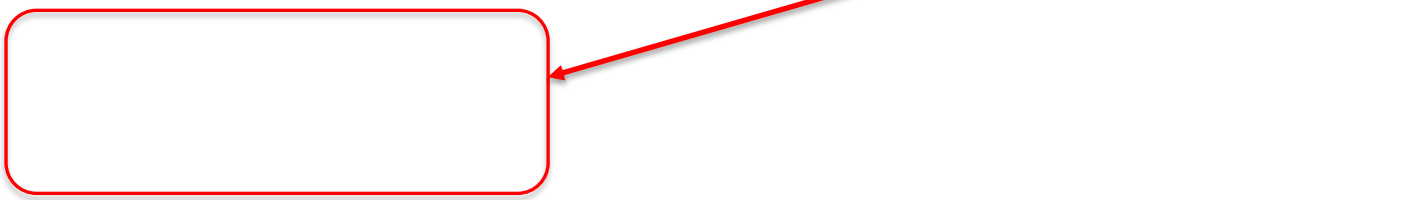
returnwaarde van Subscribe() is een object afgeleid van IDisposable: de unsubscriber.

Een observer kan de unsubscriber gebruiken om zijn abonnement stop te zetten.

(zie voorbeeldcode op msdn)

Observer Pattern in .NET

IObserver<T>



Observer Pattern in .NET

IObserver<T>: methodes

- OnNext()
  - komt overeen met update() uit het pattern
  - pusht een T-object naar de observer

- OnError()
  - signaleert de observer dat het subject het noorden kwijt is

- OnComplete()
  - signaleert de observer dat het subject geen data meer zal doorgeven

Observer Pattern
Oefeningen

- Implementeer je eigen weerstation, in java of in een .NET-taal. Een weerstation bestaat uit een thermometer en displays. De thermometer stuurt op regelmatige tijden de huidige temperatuur naar de displays.

- Bij het programmeren van GUI's worden dikwijls events gebruikt (bijvoorbeeld: wanneer de gebruiker op een button klikt, dan krijg je een event).
  Is dat een toepassing van het Observer Pattern? Waarom wel of waarom niet?

Observer Pattern
Oefeningen

- In een treinstation komen treinen aan en vertrekken ze weer.
  Een aankomst gaat over een bepaald treinnummer, een spoornummer, en een tijd van aankomst.
  Een vertrek gaat over een bepaald treinnummer, een spoornummer, en een tijd van vertrek.
  Schrijf een systeem dat het observer pattern gebruikt, om informatie te tonen over de treinen in het station.