HET OBSERVER PATTERN

Veerle Ongenae, Wijnand Schepens

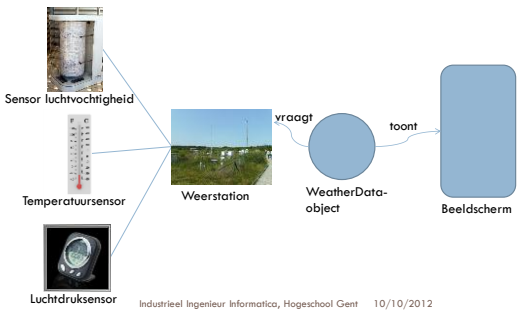
Het Observer Pattern

- Hou je objecten geïnformeerd
- Objecten kunnen at runtime besluiten of ze geïnformeerd blijven
- Veel gebruikt in de JDK
- Een-op-veelrelaties
- Zwakke koppeling

Voorbeeld

- Internet weerstation
  - Huidige situatie
    - Temperatuur
    - Vochtigheid
    - Luchtdruk
  - Weerstatistieken
  - Weersverwachting
- Uitbreidbaar zijn
  - Eigen weerschermen door andere ontwikkelaars

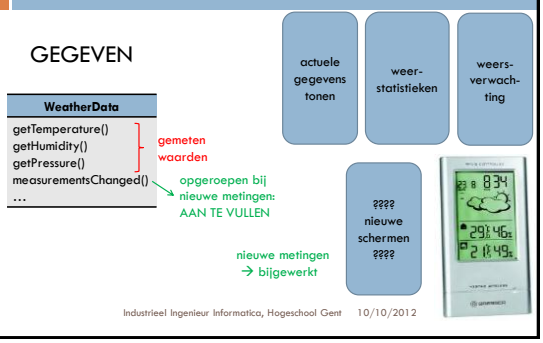
Voorbeeld: gegeven



Voorbeeld

GEGEVEN

TO DO



Eerste poging


```
public class WeatherData {
    …
    public void measurementsChanged() {
        float temp     = getTemperature();
        float humidity = getHumidity();
        float pressure = getPressure();

        currentConditionDisplay.update(temp, humidity, pressure);
        statisticsDisplay.update(temp, humidity, pressure);
        forecastDisplay.update(temp, humidity, pressure);
    }
    …
}
```

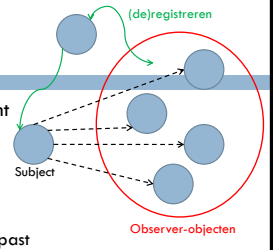
Wat is er verkeerd?

7

- Programmeren naar concrete implementaties
  - Code moet aangepast om schermen toe te voegen of te verwijderen
- Het deel dat verandert is niet ingekapseld

Observer Pattern

8

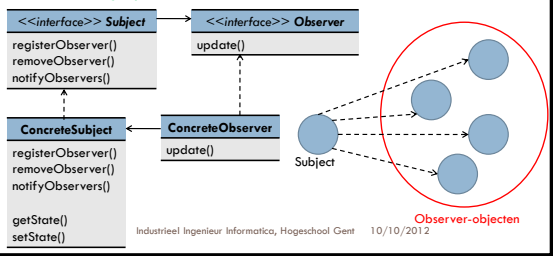


- Principe krantenabonnement
  - Subject = uitgever
  - Observer = abonnee
- Principe
  - Subject verwerkt gegevens
  - Gegevens in Subject aangepast
    - Observers krijgen bericht
    - Nieuwe gegevens worden naar de Observer gecommuniceerd
  - Observers zijn geregistreerd bij het Subject (hebben een abonnement)
    - Registratie kan toegevoegd of verwijderd worden

Observer Pattern: definitie

9

- Het Observer Pattern definieert een een-op-veel-relatie tussen objecten, zodanig dat wanneer de toestand van een object verandert, alle afhankelijke objecten worden bericht en automatisch worden geüpdatet.



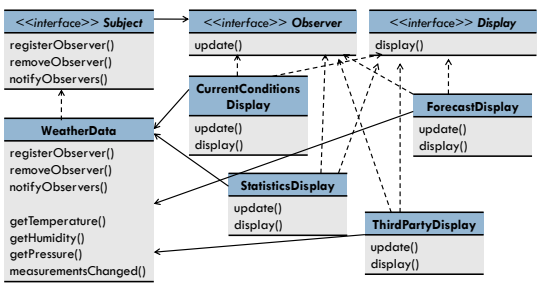
Zwakke koppeling

10

- Observer Pattern gebruikt een zwakke koppeling
- Zwakke koppeling tussen objecten
  - Interactie mogelijk
  - Weinig info over elkaar
- Ontwerpprincipe
  - Streef naar ontwerpen met een zwakke koppeling tussen de objecten die samenwerken
- Zwakke koppeling → flexibele systemen
  - Eenvoudig veranderen
  - Wederzijdse afhankelijkheid is klein

Weerstation en Observer Pattern

11



Weerstation: interfaces

12

```
public interface Subject {
     public void registerObserver(Observer o);
     public void removeObserver(Observer o);
     public void notifyObservers(Observer o);
}

public interface Observer{
     public void update( float temp, float humidity, float
   pressure);
}

public interface Display {
     public void display();
}
```

Weerstation: WeatherData

13

```
class WeatherData  implements Subject {
    private float temperature, humidity, pressure;
    private ArrayList<Observer> observers;
    …

    public void measurementsChanged() {
         notifyObservers();
    }
    public void setMeasurements(…) {
        …
        measurementsChanged();
    }
```

```
    public void registerObserver( Observer o) {
        observers.add(o);
    }

    public void removeObserver( Observer o) {
         observers.remove(o);
    }

    public void notifyObservers( ) {
          for (Observer o : observers)
               o.update(temperature, humidity, pressure);
    }
    …
}
```


Weerstation: schermen

14

```
class CurrentConditionsDisplay implements Observer, Display {
    private float temperature, humidity;
    private Subject weatherData;
    public CurrentConditionsDisplay (Subject weatherData) {
         this.weatherData = weatherData;
         weatherData.registerObserver(this);
    }

    public void update(float temp, float humidity, float pressure) {
         this.temperature = temp;
         this.humidity = humidity;
         display();
    }
    public void display() { ... }
}
```


Informatie uitwisselen

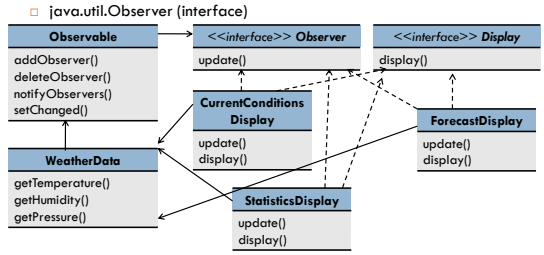
15

- Weerstation
  - Informatie via argumenten update-methode
- Twee mogelijkheden
  - Push-model
    - Subject stuurt informatie naar observers
  - Pull-model
    - Subject stuur minimaal bericht
    - Observers vragen om details
- Andere naam Observer Pattern
  - Publish-Subscribe

Observer Pattern van Java

16

- java.util.Observable (klasse)
- java.util.Observer (interface)



Anders?

17

- Observable is een klasse
  - WeatherData erft van Observable
- Observable berichten laten sturen
  - setChanged() aanroepen
  - notifyObservers() of notifyObservers(Object arg) aanroepen
- Bericht ontvangen
  - update(Observable bron, Object arg) implementeren
    - Push-methode: info in arg
    - Indien observer van meerdere subjects: kijk naar bron

Weerstation: WeatherData (versie 2)

18

```
public class WeatherData extends Observable {
    private float temperature, humidity, pressure;

    public void measurementsChanged() {
         setChanged();
         notifyObservers();
    }
    public float getTemperature() {
         return temperature;
    }
    // ...
    public void setMeasurements(...) {
        ...
        measurementsChanged();
    }
```

Weerstation: schermen (versie 2)

19

```
public class CurrentConditionsDisplay implements Observer, Display {
    private float temperature, humidity;
    private Observable weatherData;
    public CurrentConditionsDisplay (Observable weatherData) {
        this.weatherData = weatherData;
        weatherData.addObserver(this);
    }
    public void update(Observable obs, Object arg) {
        if (obs instanceof WeatherData) {
            WeatherData data = (WeatherData) obs;
            this.temperature = obs.getTemperature();
            this.humidity = obs.getHumidity();
            display();
        }
    } … }
```


Opmerking

20

- De volgorde waarin de observers een bericht krijgen is onbepaald.

Nadelen java.util.Observable

21

- Observable is een klasse, geen interface
  - Jouw klasse kan dus niet afgeleid zijn van een andere klasse
- Observable implementeert geen interface
  - Je kan geen eigen implementatie maken
  - Je kan de implementatie niet vervangen door een andere
- setChanged() is protected
  - Moet afgeleide klasse maken
  - Strijdig met “Prefereer compositie boven overerving”

Observer Pattern in JDK

22

- JavaBeans
- Swing
  - Luisteraars
    - ActionListener
    - …
- …

Samenvatting kennis

23

- OO-basis
- OO-principes
  - Isoleer wat verandert
  - Verkies compositie boven overerving
  - Programmeer naar een interface en niet naar een implementatie
  - Streef naar ontwerpen met een zwakke koppeling tussen interagerende objecten
- OO-patterns
  - Strategy
  - Observer: definieert een een-op-veelrelatie tussen objecten, zodat wanneer de toestand van één object verandert, alle afhankelijke objecten bericht ontvangen en automatisch worden bijgewerkt.