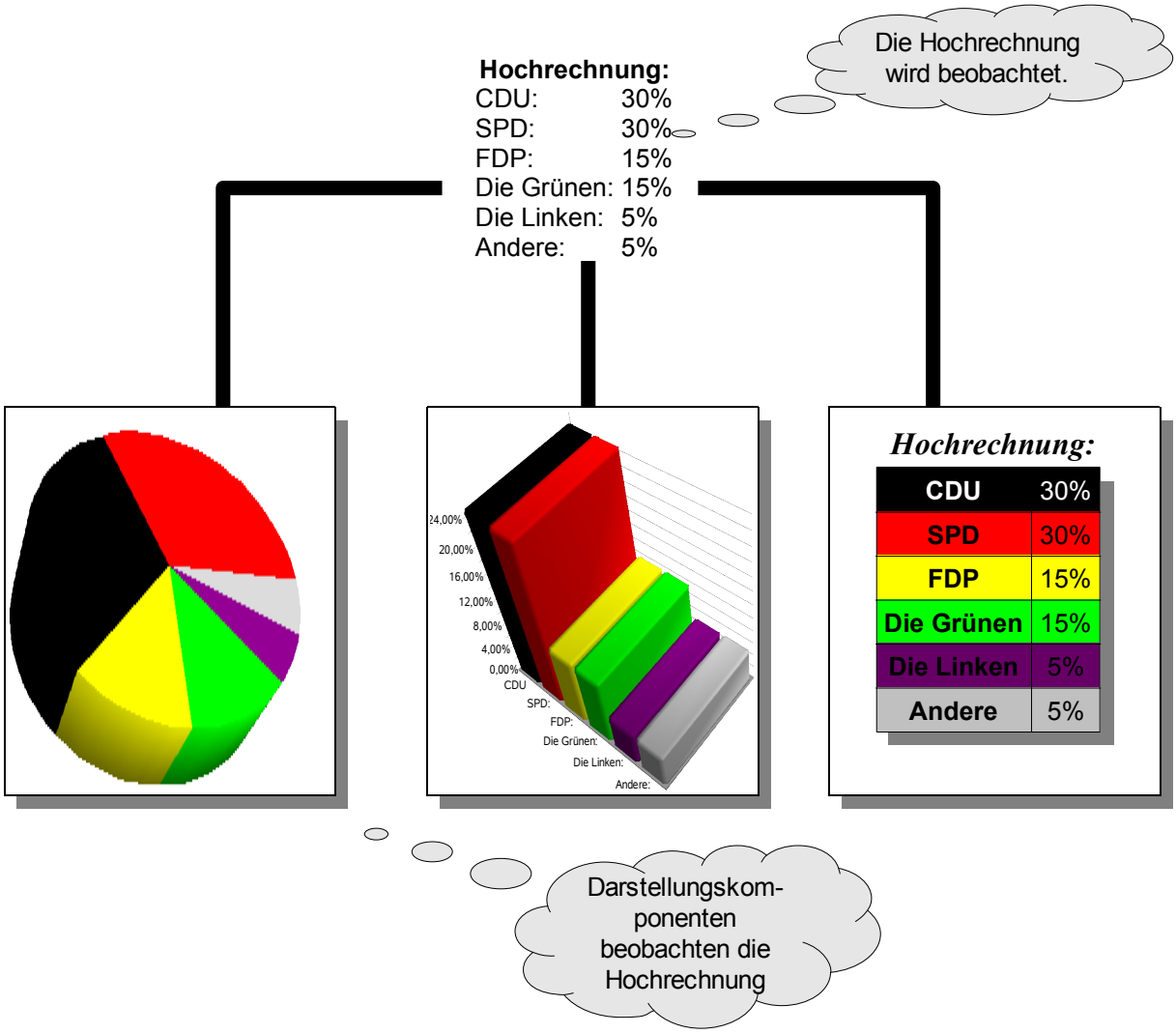
Beobachter (Observer)

Der Observer (Beobachter) ist ein Entwurfsmuster aus dem Bereich der Softwareentwicklung und gehört zu der Kategorie der Verhaltensmuster (Behavioural Patterns). Es dient zur Weitergabe von Objekt-Änderungen an von diesem Objekt abhängige Strukturen. Das Muster ist eines der sogenannten GoF-Muster (Gang of Four).

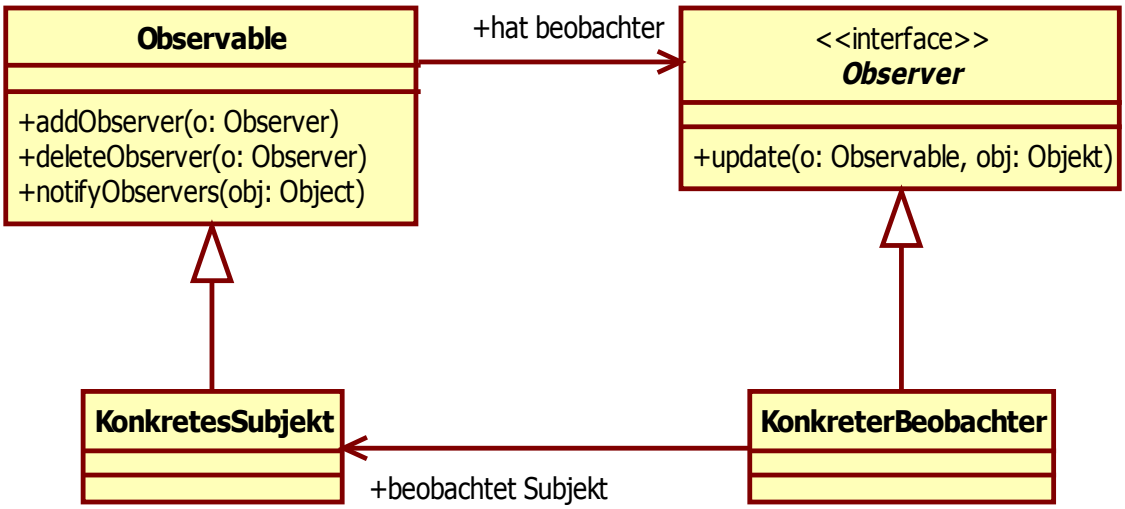
Anwendungsbeispiel

Eine oder auch mehrere Komponenten stellen den Zustand eines Objektes grafisch dar. Sie kennen die gesamte Schnittstelle dieses Objektes. Ändert sich der Zustand des Objektes, müssen die Komponenten darüber informiert werden. Andererseits soll das Objekt aber von den Komponenten unabhängig bleiben - ihre Schnittstelle also nicht kennen.
Beispiel: Hochrechnungen von Wahlergebnisse werden gleichzeitig in einem Balkendiagramm, einem Tortendiagramm und einer Tabelle dargestellt. Hochrechnungen ändern sich permanent. Die Komponenten der Diagramme sollen diese Änderungen permanent darstellen. Die eigentliche Hochrechnung soll aber möglichst unabhängig von den Darstellungskomponenten und deren Struktur bleiben.



Lösung

Das beobachtete Objekt (`Observable`) bietet einen Mechanismus, beispielsweise eine verkettete Liste, um Beobachter / Observer an- und abzumelden (`addObserver()`, `deleteObserver()`) und diese über Änderungen zu informieren (`notifyObservers()`). Es kennt alle seine Beobachter / Observer nur über die Schnittstelle `Observer`, weiß also lediglich, dass alle Beobachter / Observer die Methode `update()` implementiert haben müssen. Es meldet jede Änderung völlig unspezifisch an jeden angemeldeten Observer, braucht also die weitere Struktur dieser Komponenten nicht zu kennen. Die Beobachter implementieren ihrerseits eine Methode (`update()`), um auf die Änderung zu reagieren.



Lösung in Java

In Java müssen die Klassen `Observable` und auch das Interface `Observer` nicht selbst erstellt werden. Stattdessen kann auf die Komponenten `java.util.observer` und `java.util.Observable` zurückgegriffen werden. Zudem bietet Java mit dem Event-Handling-Konzept eine implizite Observer (Listener)-Implementierung für Eingabeereignisse an.

Witzeerzähler

In dem folgenden sehr einfachen Beispiel wird ein Witzeerzähler von zwei Zuhören beobachtet.

Die main-Funktion

In der **main**-Funktion werden ein Witzeerzähler und zwei Zuhörer erzeugt. Die Zuhörer beobachten zunächst beide den Witzeerzähler und lachen über den Scherz. Anschließend verliert ein Zuhörer das Interesse an den niveaulosen Sprüchen.

```
package observer;

public class Party
{
   public static void main(String[] args)
   {
   // Objekte generieren
      Witzeerzaehler    erzaehler_uwe  = new Witzeerzaehler("Uwe");
      Zuhoerer       zuhoerer_eva      = new Zuhoerer ("Eva");
      Zuhoerer       zuhoerer_kai      = new Zuhoerer ("Kai");

   // Beobachter registrieren
      erzaehler_uwe.addObserver(zuhoerer_eva);
      erzaehler_uwe.addObserver(zuhoerer_kai);

   // Statusenderung
      erzaehler_uwe.erzaehleWitz ("Eine Null kann bestehende Probleme verzehnfachen");

   // Beobachter löschen
      erzaehler_uwe.deleteObserver(zuhoerer_eva);

   // Statusaenderung
      erzaehler_uwe.erzaehleWitz ("Was ist ein Held ohne Geld?");
   }
}
```

Bildschirmausgabe

```
**************************************
Die Person: Kai
lacht über den Witz: Eine Null kann bestehende Probleme verzehnfachen
von der Person : Uwe
**************************************
Die Person: Eva
lacht über den Witz: Eine Null kann bestehende Probleme verzehnfachen
von der Person : Uwe
**************************************
Die Person: Kai
lacht über den Witz: Was ist ein Held ohne Geld?
von der Person : Uwe
```

Beobachtbar (Observable)

Der Witzeerzähler kann beobachtet werden. Jedes Mal, wenn er einen neuen Spruch ablässt **`erzaehleWitz()`**, muss er alle seine Zuhörer über diese Zustandsänderung in Kenntnis setzen **`notifyObserver()`**.

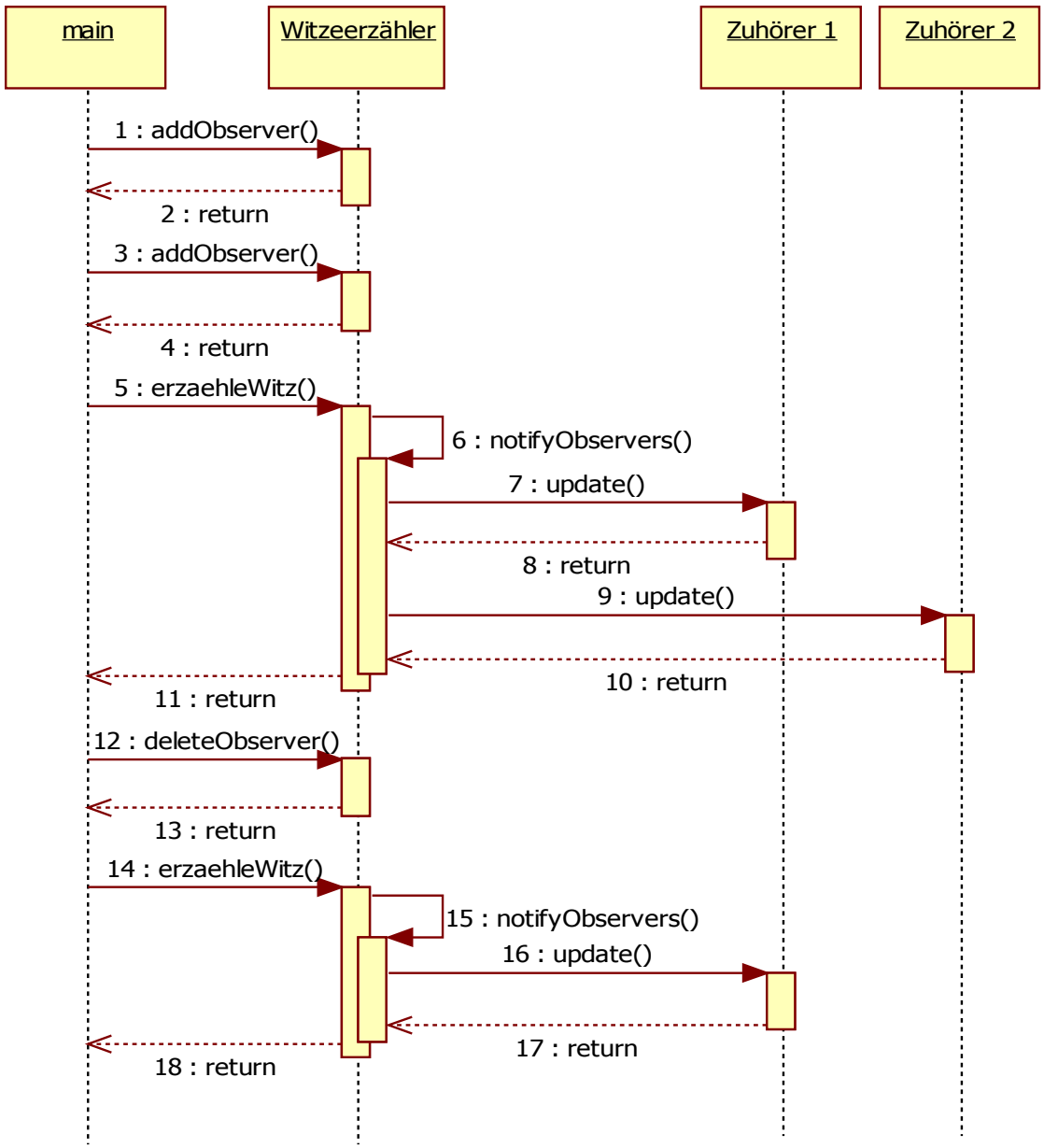
```
package observer;
import java.util.Observable;

public class Witzeerzaehler extends Observable
{
   String nameErzaehler;

   public Witzeerzaehler (String nameErzaehler)
   {
      this.nameErzaehler = nameErzaehler;
   }

   public String getName ()
   {
      return this.nameErzaehler;
   }

   public void erzaehleWitz (String witz)
   {
      setChanged();           // Signalisiert Zustandsänderung
      notifyObservers(witz);        // Benachrichtigung an alle Beobachter
   }
}
```

Beobachter (Observer)

Es muss bei dem Beobachter eine Methode **`update()`** existieren, über die das beobachtete Objekt über Statusänderungen informieren und gegebenenfalls diese Statusänderungen auch weiterreichen kann. Die Methode **`update()`** ist die einzige Methode des Interfaces **`Observer`**, das von den Beobachtern implementiert werden muss.

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

public class Zuhoerer implements Observer
{
   String nameZuhoerer;

   public Zuhoerer (String nameZuhoerer)
   {
      this.nameZuhoerer = nameZuhoerer;
   }

   public void update (Observable o, Object obj)
   {
      String nameErzaehler    = ((Witzeerzaehler) o).getName();
      String witz          = (String)obj;

      System.out.println ( " ************************************ ");
      System.out.println ( " Die Person: "         + nameZuhoerer);
      System.out.println ( " lacht über den Witz: "   + witz);
      System.out.println ( " von der Person : "       + nameErzaehler);
   }
}
```

Sequenzdiagramm Witzeerzähler



Anmerkung

In der von Java vorgegebenen Lösung wird durch den Funktionsaufruf **`update()`** eine Referenz des beobachteten Objektes und ein Aufrufobjekt gleich mit übergeben, so dass die Beobachter sofort über die Änderungen bzw. über den neuen Zustand des beobachteten Objektes (hier Witzeerzähler) informiert sind.
In vielen anderen Umsetzungen werden die Beobachter lediglich über die Methode **`update()`** informiert, das eine Statusänderung vorliegt. In einem solchen Fall müssen sich die Beobachter den veränderten Status erst noch bei dem beobachteten Objekt abholen. Dies kann zum Beispiel über eine Methode **`gibStatus()`** des beobachteten Objektes erfolgen.