Design-Pattern Observer

Observer1.java

Beispiel für das Observer Design Pattern mit einem einfachen Dialogfenster
Mit der Eingabetaste wird der Inhalt aktualisiert

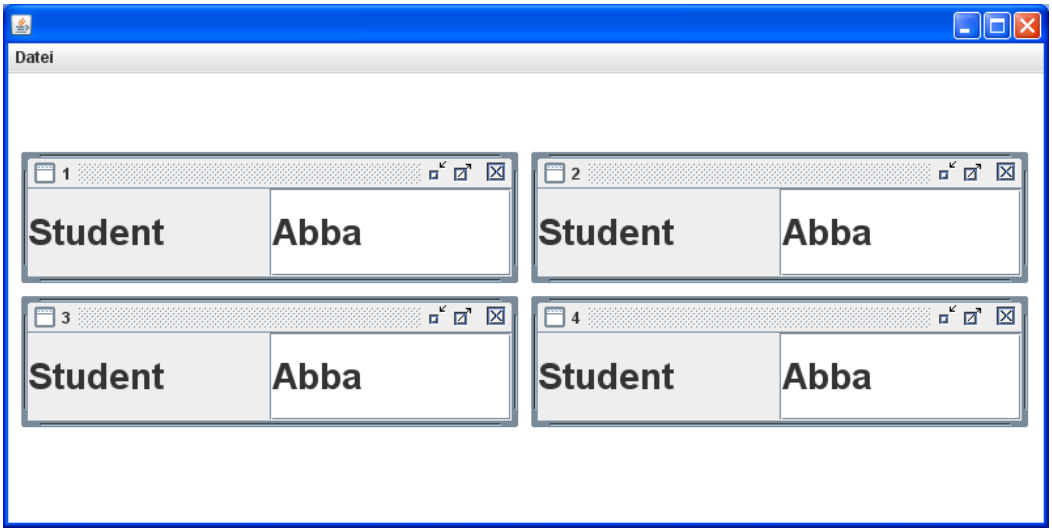

Abbildung 1 Observer1.java

Lösung: Modell und Viewer in Extra-Klassen

Mit der Eingabetaste werden die Daten in den Fenster abgeglichen.

```
// 1. Schritt Datenmodell erstellen
// 2. Schritt Controller ableiten von Observable
// 3. Schritt: Im Controller eine Methode für die Viewer zum Updaten schreiben

// 4. Schritt  die Modell Variable im JFrame deklarieren
// 5. Schritt  Modell Variable erzeugen
// 6. Schritt  Controller Variable erzeugen

// 7. Schritt  im ÍnternalFrame  Implements einbauen und die Methode Update
        public void update(Observable o, Object arg) {  }

// 8. Schritt  Methode edit_change (Enter) implementieren, Aufruf des Controllers
// 9. Schritt  Viewer in der Methgode „New_click“ erzeugen
```

Observer2.java

Funktion wie Beispiel im Observer1.java

Die Änderung finden aber sofort statt (DocumentListener mit JTextFieldDocListener)

dazu wird eine neue Klasse „JTextFieldDocListener“ von JTextField abgeleitet
Diese implentiert DocumentListener und fängt die Change-Events ab
Danach werden diese an das JInternalFrame weitergeleitet
Dies geschieht durch ein Interface ActionPerformed

```
// 1. Import
import java.util.Observable;
import java.util.Observer;

// 2. Schritt ableiten von Observable, damit Modell und Obervable
class Modell  extends java.util.Observable {

   // 3. Schritt: Im Controller eine Methode für die Viewer zum Updaten schreiben
  public void DataChangedFromViewer(int sourceViewerTag, String s) {

   // 4. Schritt  die Modell Variable im JFrame deklarieren
  private Modell modell;   // Modell

    // 5. Schritt  die Modell Variable „erzeugen“: Datei laden etc.
    modell = new Modell();
    modell.setName("Paul");

// 6. Schritt  im JFrame  Implements einbauen
class JObserverFrame extends JFrame implements Observer, ActionListener{

  // 7. Schritt  Modell deklarieren
  private Modell modell;

  // 8. Schritt  Konstruktor
  public JObserverFrame( String caption, Observable datenModell) {

    // 9. Schritt  globale Variable setzen
    modell = (Modell) datenModell;  //

    // 10. Schritt  im JFrame  registrierung
    modell.addObserver(this);

    // 11. Schritt  aus dem Modell die Daten holen
    modell.setUpdateModus(true);  // hier wird aktualisiert, sperre
    edit.setText(  modell.getName()     );
    modell.setUpdateModus(false);  // hier wird aktualisiert, sperre


  // 12. Schritt  Schnittstellen implementieren: Methode Update
  // aufgerufen vom Controller
    // Weiterleitung abschalten
    // in arg steht die int sourceViewerTag
    // hier muss eine rekusive Schleife verhindert werden
    // ohne die Abfrage gibt es eine Exception !
  public void update(Observable o, Object arg) {
    int sourceViewerTag = ((Integer) arg).intValue();
    System.out.println("sourceViewerTag: "+sourceViewerTag+"   tag: "+tag);
    if (sourceViewerTag != tag) {
      System.out.println("Update");
      if ( modell.getUpdateModus() ) {
        System.out.println("JIF Update: getUpdateModus: "+modell.getName());
```

```
      edit.setText( modell.getName() );
    }
  }
  else {
    System.out.println("bin ich selber sourceViewerTag");
  }
} // update


 // 14. Schritt  Methode edit_change (Enter) implementieren, Aufruf des Controllers
  // wird vom JTextFieldDocListener aufgerufen
public void actionPerformed(ActionEvent e){
```

Observer3.java

Funktion wie Beispiel Observer2.java.
Hier gibt es aber **zwei** JTextFields

Observer4.java

Funktion wie Beispiel Observer2.java.
Hier gibt es aber ein **JTextArea**