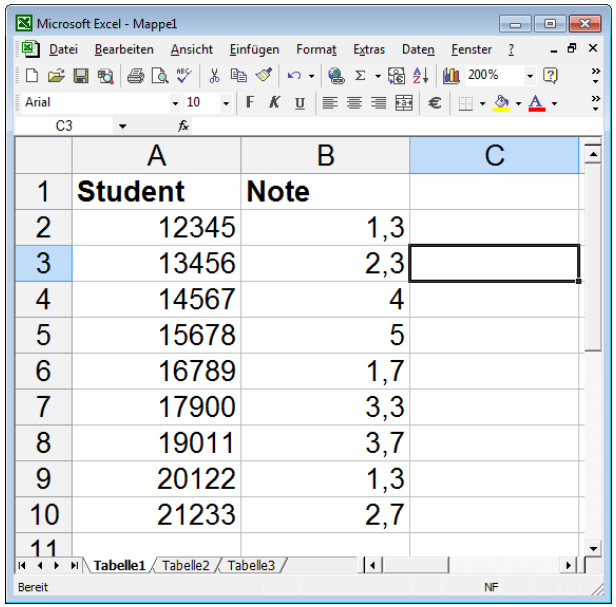
Wahlpflichtfach Design Pattern

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- miwilhelm@hs-harz.de
- http://www.miwilhelm.de
- Raum 2.202
- Tel. 03943 / 659 338

Inhalt

MDI-Fenster:

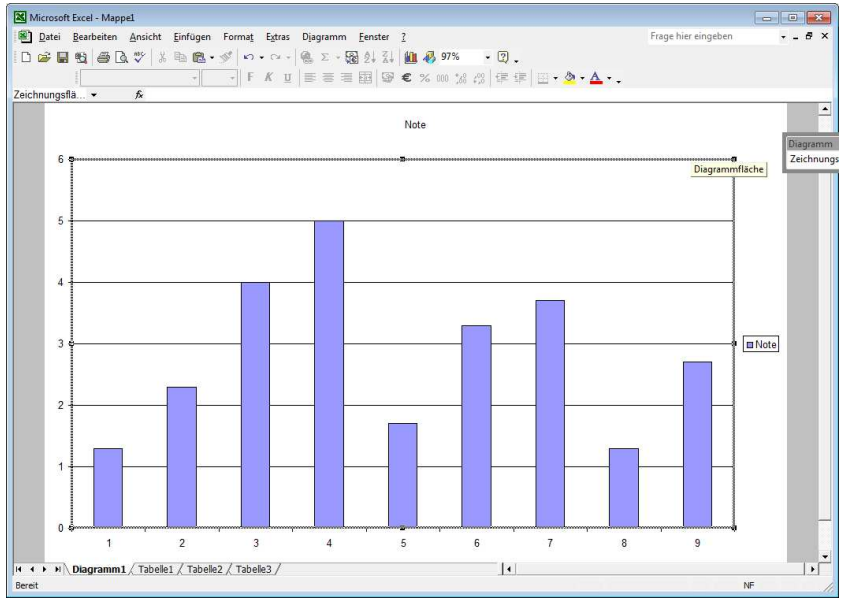
Daten in einer Tabelle

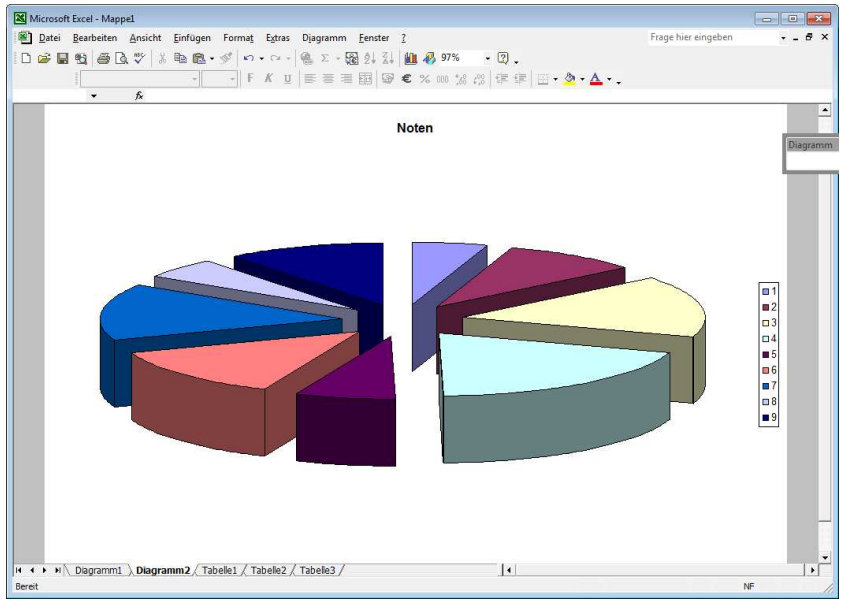


	A	B	C
1	**Student**	**Note**	
2	12345	1,3	
3	13456	2,3	
4	14567	4	
5	15678	5	
6	16789	1,7	
7	17900	3,3	
8	19011	3,7	
9	20122	1,3	
10	21233	2,7	
11			

Problem:
Datenaktualisierung

Daten in einer Grafik





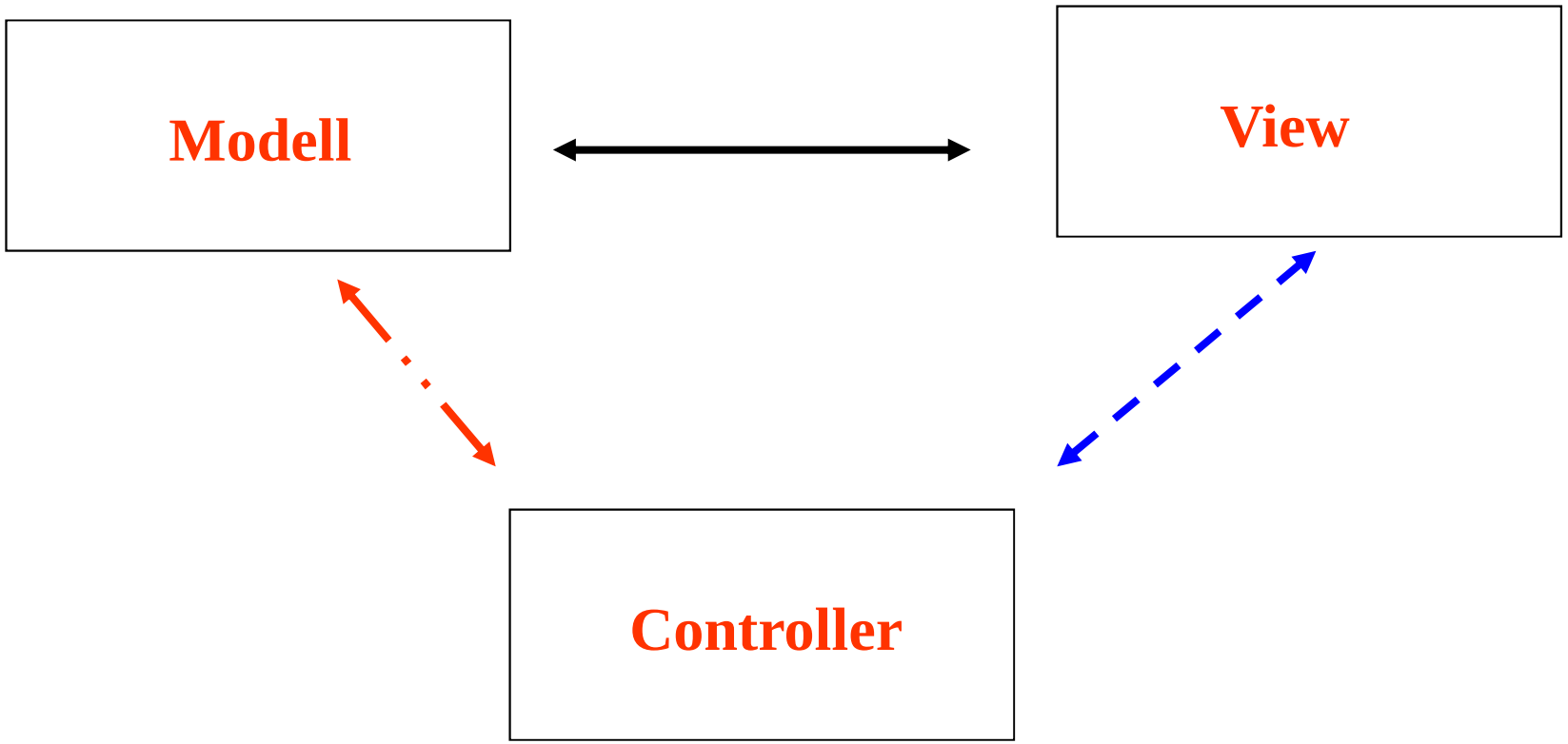
2. Beispiel: Observer Pattern

Problem:

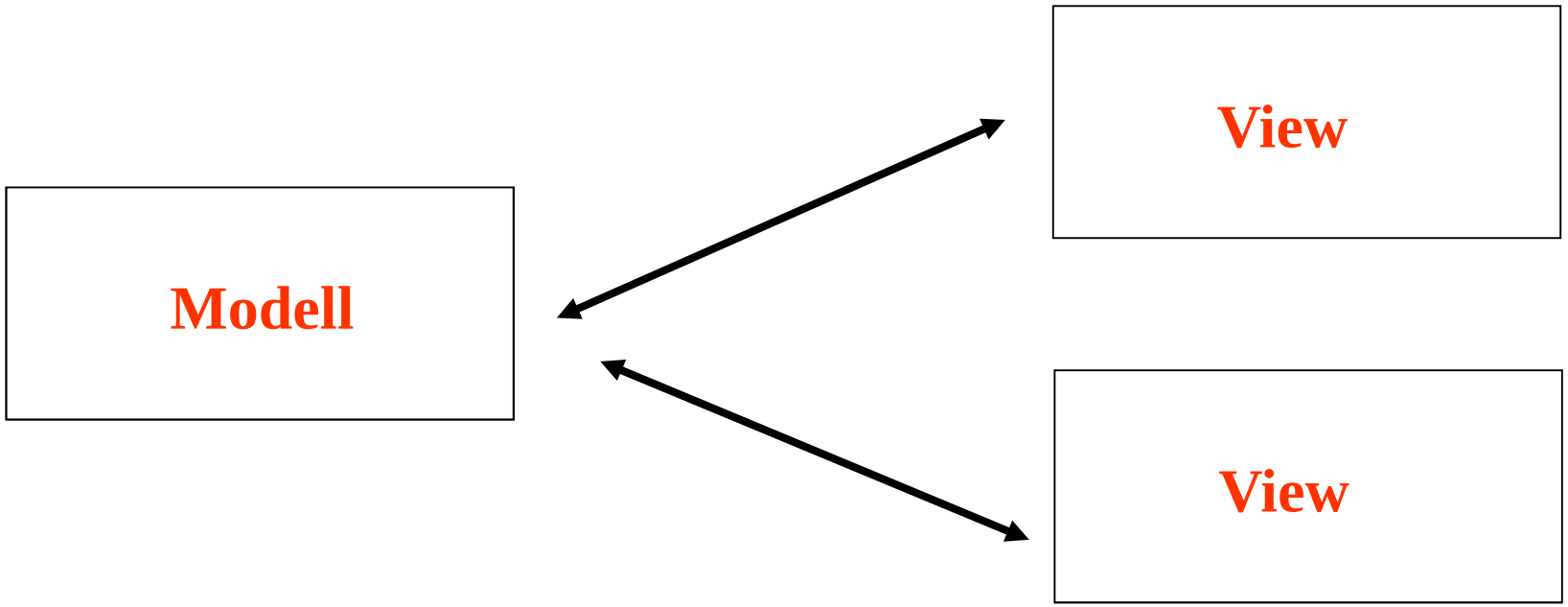
Mehrere Objekte einer Gruppe müssen sich benachrichtigen, wenn einige Attribute geändert wurden. Die Viewer sind nicht „bekannt“.

Smalltalk:	Model-View-Controller
MFC	Dokument-View-Architektur
Java:	Model-View-Architektur

Beispiel: Observer Pattern MVC



Beispiel: Observer Pattern MVC



Beispiel: Observer Pattern MVC

- Das Model kann man sich als das Datenmodell (Datei, Datenbank) vorstellen, der den aktuellen Zustand und das Verhalten des gesamten Systems repräsentiert (Datenhaltung, Anwendung)

- Der View hat die Aufgabe die Daten des Models auf irgend eine Art darzustellen. Dabei hat der View nur die Aufgabe, die Daten darzustellen (Visualisierung, Darstellung)

- Der Controller setzt die eingehenden Anforderungen (z.B. Eingaben von der Tastatur oder Maus) in Methoden um, die das Model dazu veranlassen, die Daten entsprechen zu verändern (Datenmanipulation, Steuerung)

Mozilla Firefox

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

Aktuelle Nachrichten - Bild.de | uml mvc darstellung - Google-Suche | http://www.famil...c/mvc_03uml.htm | [MVC] Darstellung im UML - php.de | MVC.pdf (application/pdf-Objekt)

www.familiesteudle.de/tutorium_MVC_Muster/patterns/mvc/mvc_03uml.htm

Unwesentliche Elemente dürfen *zunächst* überlesen werden. Müssen aber dennoch verstanden werden.

Wichtige Elemente:

- Startpunkt des Programms (Hauptmethode)
- Businessmethoden des Models
- Ereignisbehandelnde Methoden des Controllers
- Anzeigemethoden des View

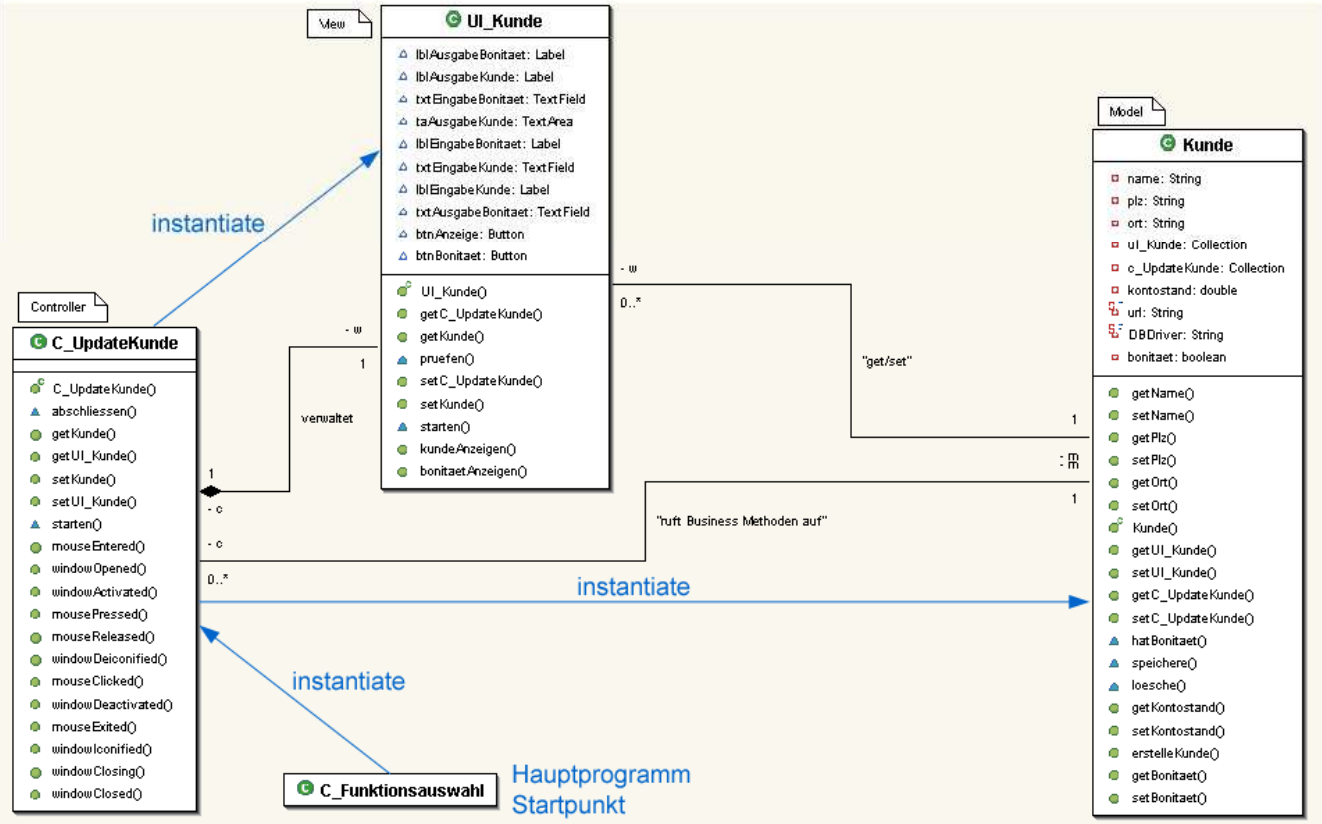
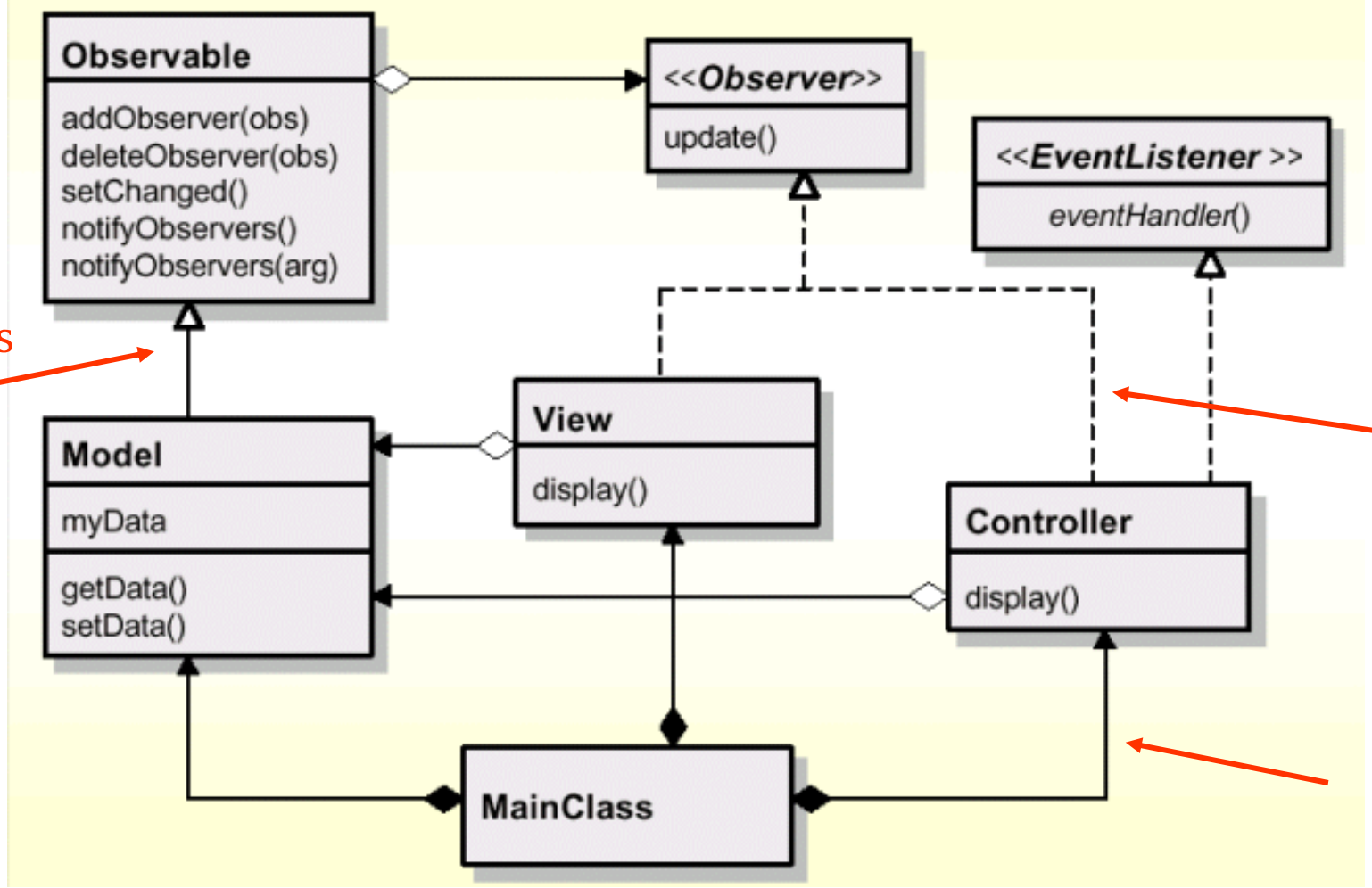


Abb. 3.4: Klassendiagramm MVC Beispiel

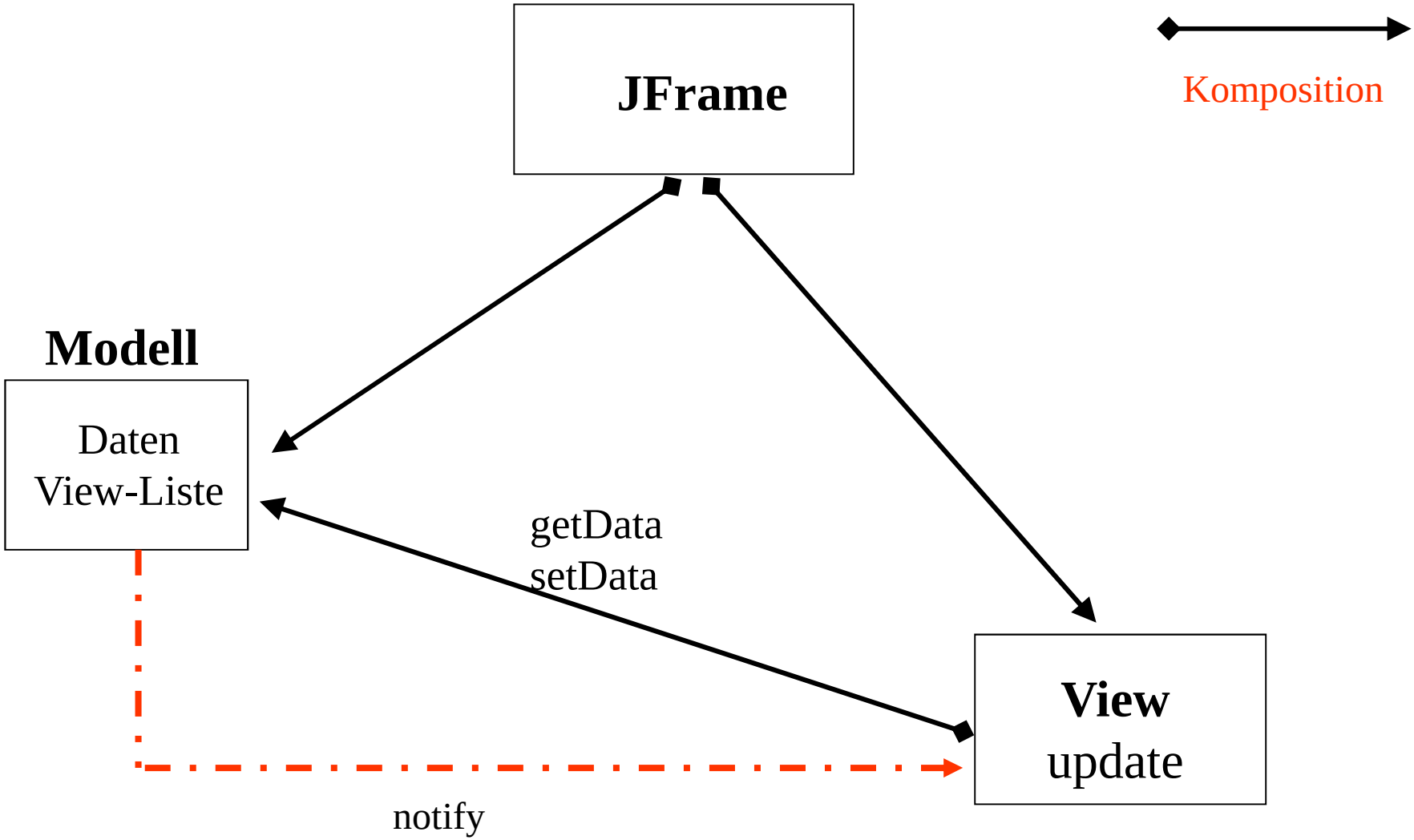
MVC mit Observable/Observer



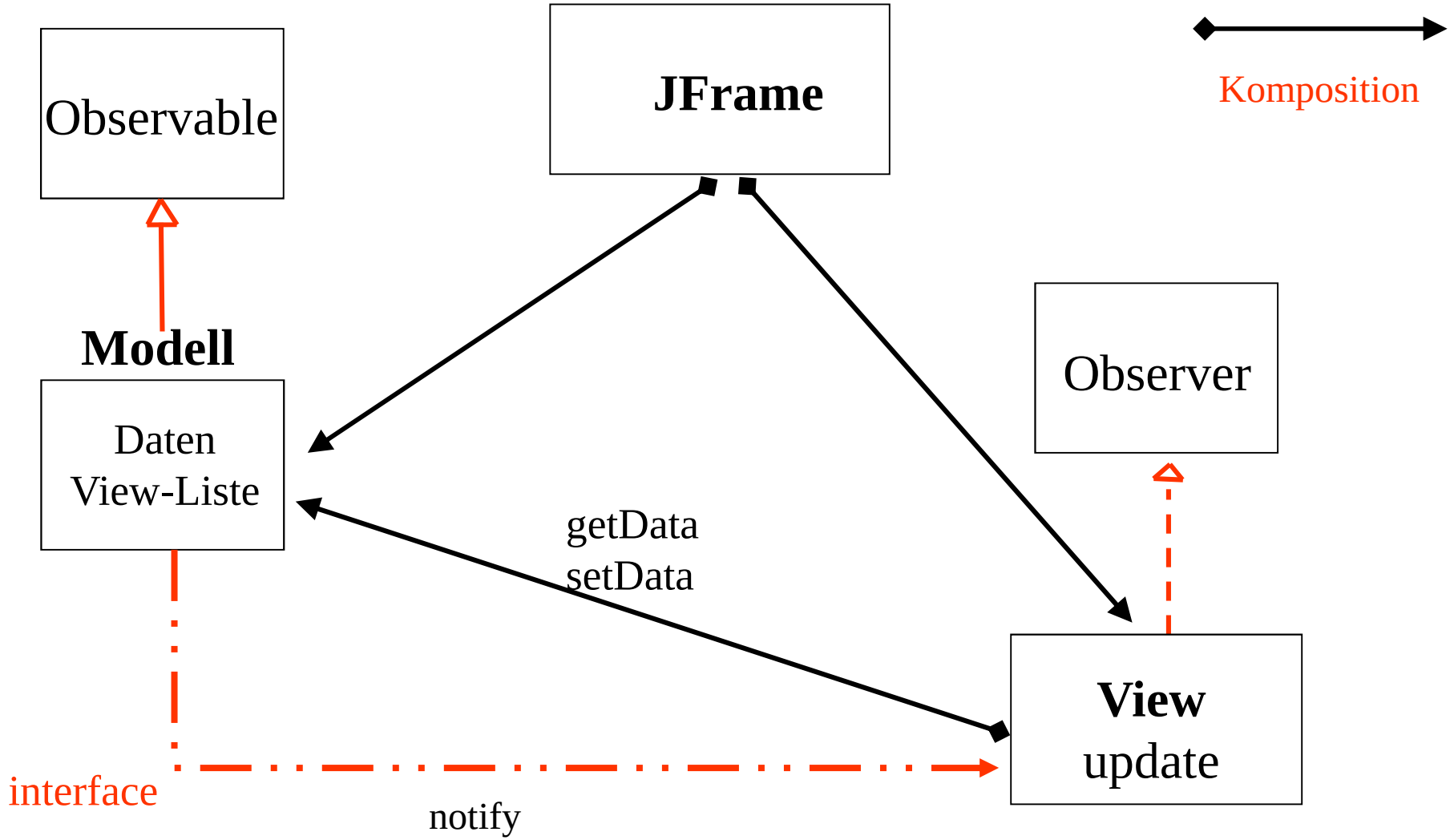
interface

Komposition

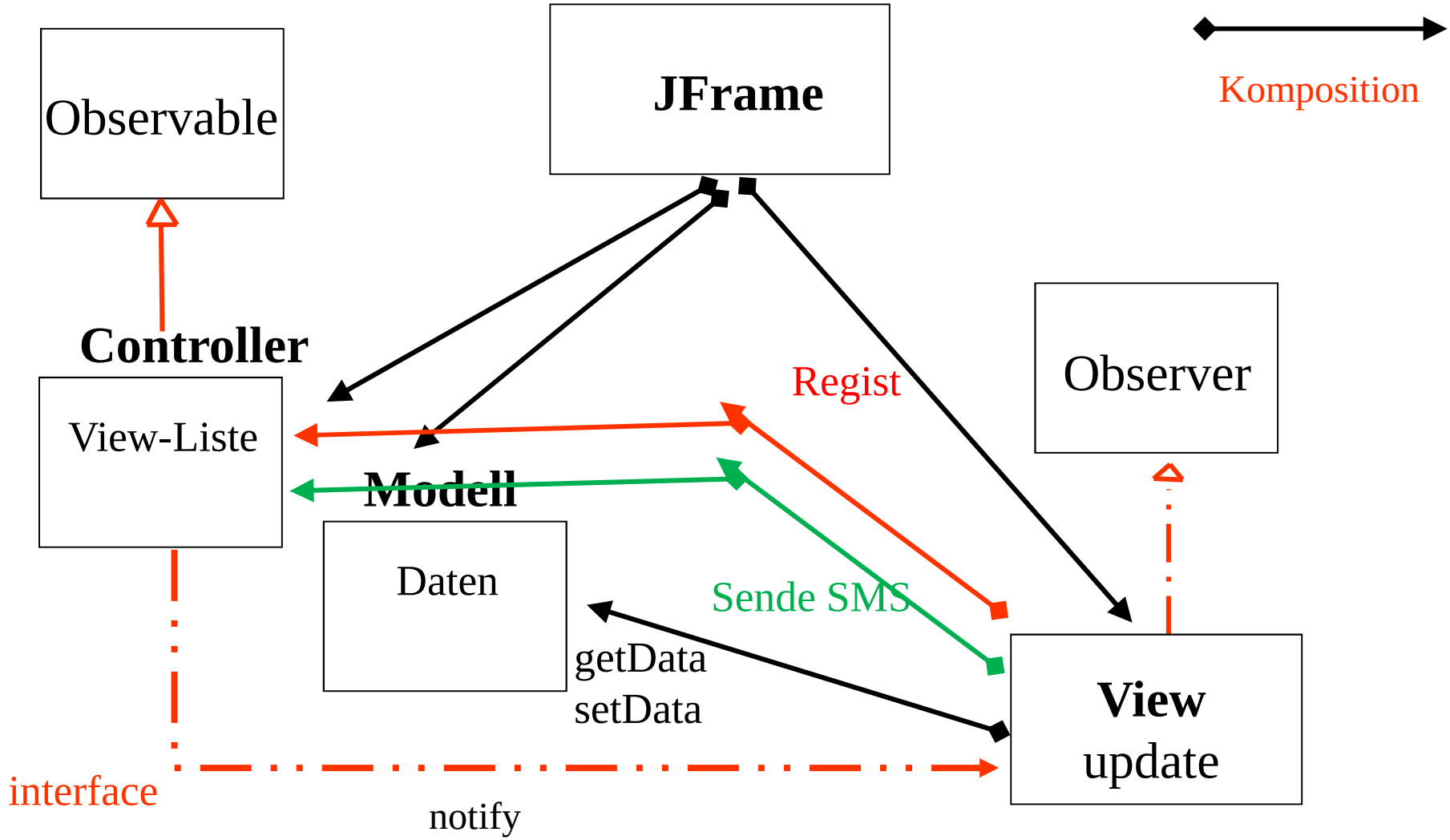
Observer Pattern (MVC ohne Controller)



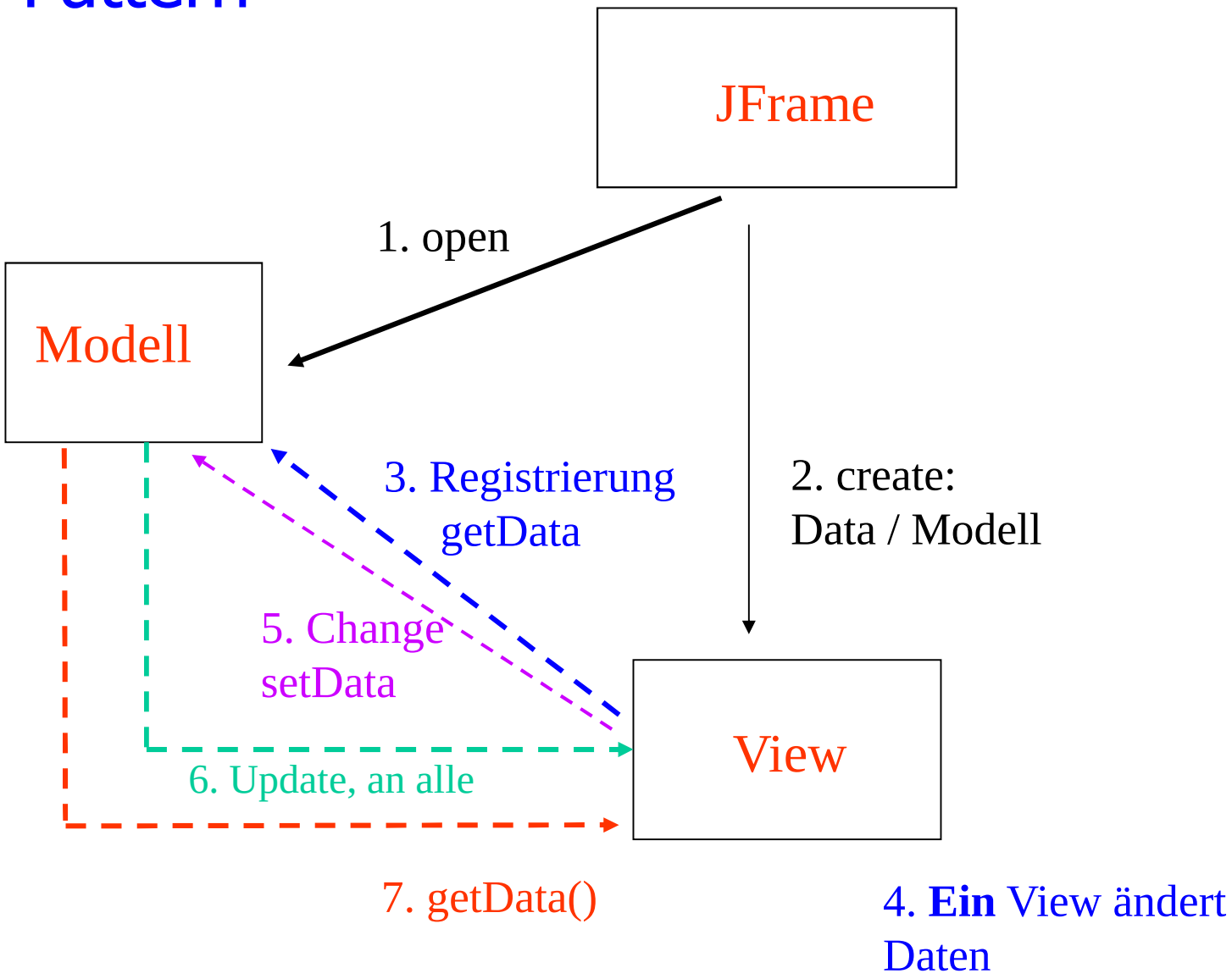
Observer Pattern (MVC ohne Controller)



Observer Pattern (MVC ohne Controller)



Observer Pattern



Bemerkungen zum Observer Pattern

- Entkopplung
  - Subjekt kennt nur Observer-Interface, keine konkreten Observer
  - update = dynamisch gebunden => upcalls
- Falls Observer von mehreren Subjekten abhängen sollen:
  - update(Observable o)
- Falls Art der Änderung übergeben werden soll: push vs pull
  - notifyObservers(Object arg)
  - update(Observable o, Object arg)
- Wann wird notify aufgerufen?
  - Zustandsänderungsfunktion => i.a. viele updates
  - Verzögert & explizit: changed-Flag: setChanged / clearChanged
    * notifyObservers wird nur aktiv wenn isChanged() == true
    * notifyObservers ruft clearChanged auf

Bemerkungen zum Observer Pattern

- Kausalität der Änderungen
  - Verbieten von Änderungen während update
  - Queueing-Mechanismus

- Multithreading
  - Während der Meldung „notify“ kann add/remove des Observers aufgerufen werden. Die Listenerliste wird geändert (InvalidStateException bei Iterator)
    * synchronized(this){
      for(int i=0; i<observers.size(); i++){…..
    * Vector v;
      synchronized(this){v = (Vector)observers.clone();}
      for(int i=0; i<v.size(); i++){…

- Interfaces vs. Classes

Java-Beispiel: Main-Frame

```
public class MVC extends JFrame {
    private Modell modell;

 public MVC() {
    modell = new Modell();
    modell.setName("Paul");
}

 void MnNew_actionPerformed(ActionEvent e) {
   MyInternalFrame1 frame = new MyInternalFrame1(modell);

   frame.setBounds( 10, 10, 250, 50);
   _desktopPane.add(frame);
  }

}
```

Java-Beispiel: Anlegen eines „Datenmodells“

```
class Modell  extends java.util.Observable {
  String sName= "Freya";

  public String getName() {
    return sName;
  }

  public void setName(String Name) {
    this.sName = Name;
  }

  public void DataChangedFromViewer(String s) {
    System.out.println("s in myData: "+s);
    this.setName(s);
    super.setChanged();
    super.notifyObservers();
  }
} // Modell
```

Java-Beispiel: Clientfenster

```
class MyInternalFrame extends JInternalFrame implements Observer {
 private Modell modell;

 public MyInternalFrame(Observable DatenModell) {

    modell = (Modell) DatenModell;
    DatenModell.addObserver(this);  // Registrierung
    edit.setText(    modell.getName()     );
}
```

Java-Beispiel: Clientfenster2

```
public void setGUI()  {
  edit.addActionListener(new java.awt.event.ActionListener() {
     //  Aufgerufen wenn Return-Taste
    public void actionPerformed(ActionEvent e) {
     edit_Change(e);
    }
  });
 }  // setGUI

 public void update(Observable o, Object arg) {
  System.out.println("Update");
  edit.setText( modell.getName() );
 }

void edit_Change(ActionEvent e) {
  System.out.println("geändert");
  modell.DataChangedFromViewer( edit.getText()  );
 }
```