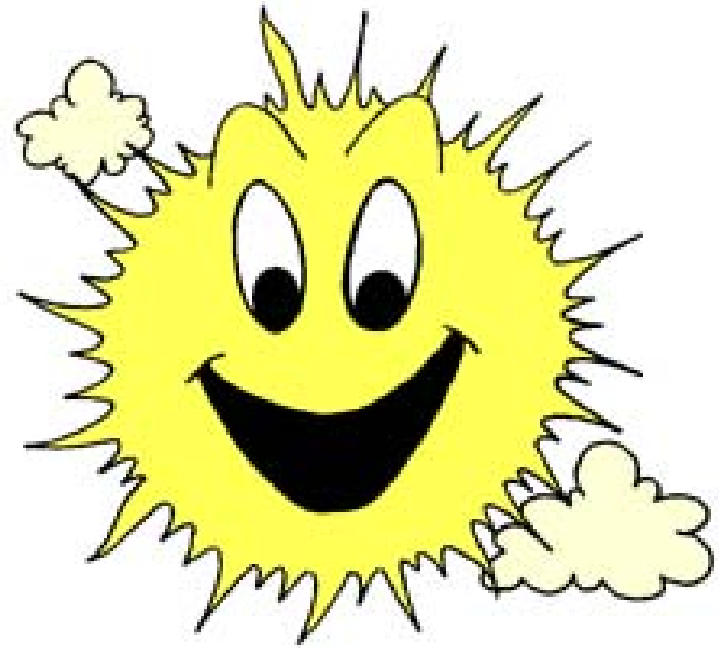
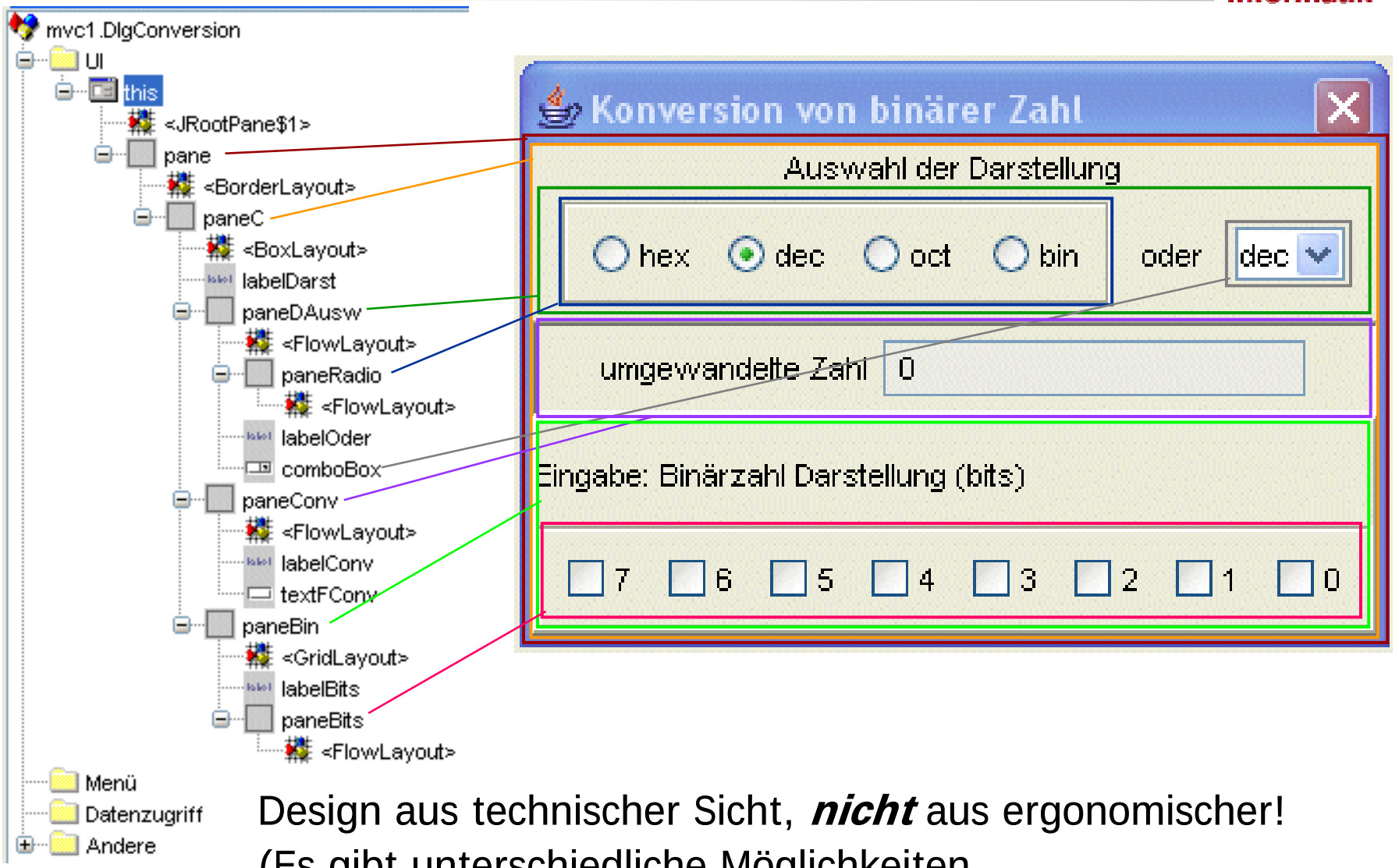
Wiederholung 6. Vorlesung



5. Layout Manager in Java

- Flow Layout
- Border Layout
- Box Layout
- Grid Layout
- Grid Bag Layout
- ...

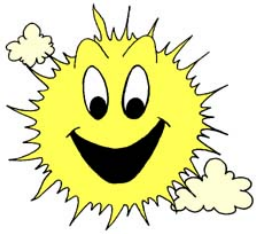
6.1 Beispiel „mvc1“ techn. Design (DlgConv)



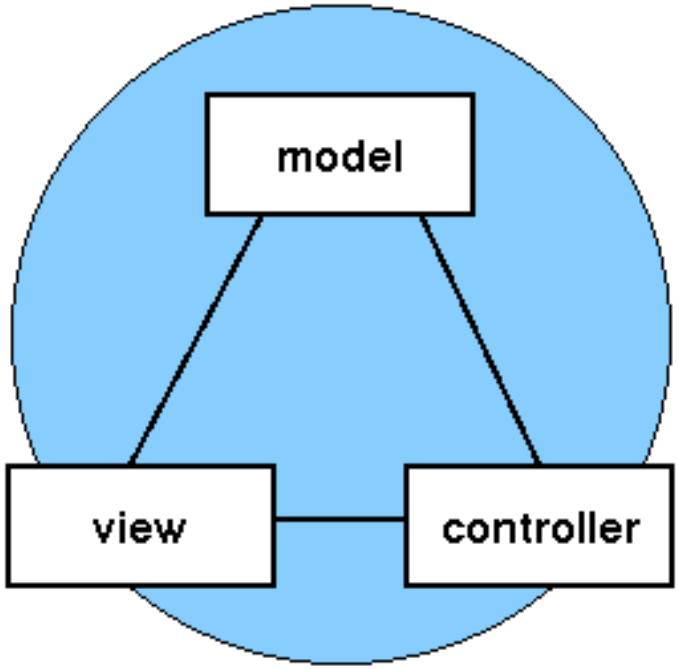
Design aus technischer Sicht, ***nicht*** aus ergonomischer!

(Es gibt unterschiedliche Möglichkeiten,
ein solches Fenster zu realisieren)

Ende der Wiederholung



6.2 Model – View – Controller



- MVC (Model – View – Controller) ist ein bewährtes Muster für Anwendungen
- MVC wurde im Lauf der Zeit weiterentwickelt (geändert)
- Es gibt eine Vielzahl Frameworks, die MVC unterstützen
- Meist leicht unterschiedliche Umsetzung

6.2.1 **Model** – View – Controller

Model:

- repräsentiert die Daten, z.B. Unternehmensdaten (DB)
- Geschäftslogik
- Verhalten und Zustand der Daten (bezüglich der Anwendung)
- kennt *nicht* spezielle Eigenschaften der Anwendung und der Visualisierungselemente
- verwaltet Verknüpfungen zu Views
- benachrichtigt Views über Änderung der Daten oder des Zustands

6.2.2 Model – **View** – Controller

Informatik

View

- spezifisch für Anwendung
- stellt über Text, Grafik oder Bitmap den Zustand des Models dar
- gibt den Inhalt des Models wieder
- weiss wie das darzustellende Gerät zu steuern ist

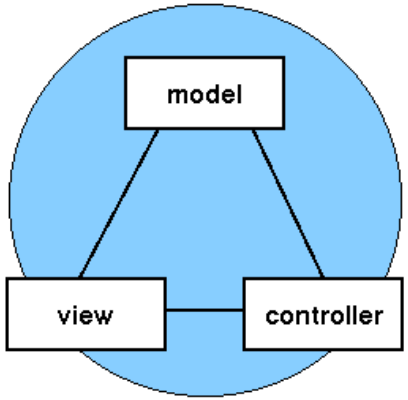
6.2.3 Model – View – **Controller**

Informatik

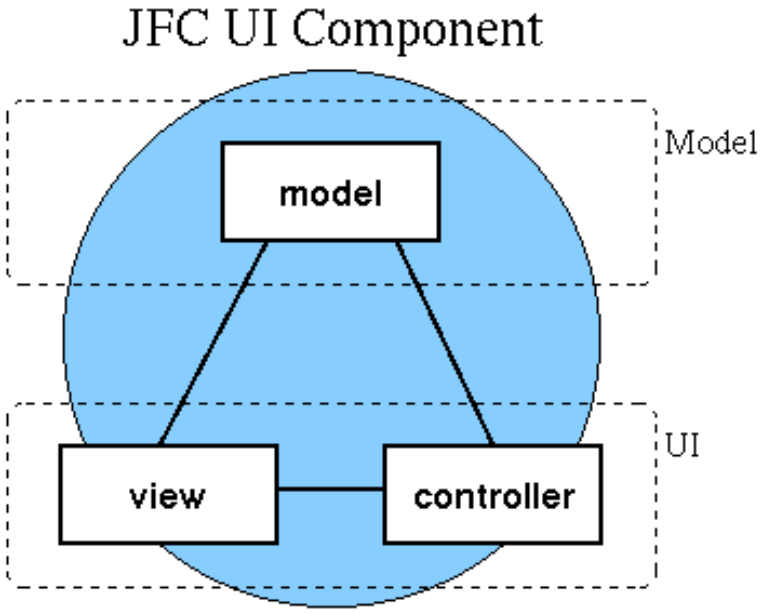
Controller

- ermöglicht die Änderung des Zustands des Models
- interpretiert Maus- und Tastatureingaben
- akzeptiert Benutzereingaben und steuert Model und View entsprechend
- verantwortlich für Umsetzung der Aktionen des Benutzers in die Antwort der Anwendung
- managt Benutzereingaben mit dem Model (Daten, Zustand)

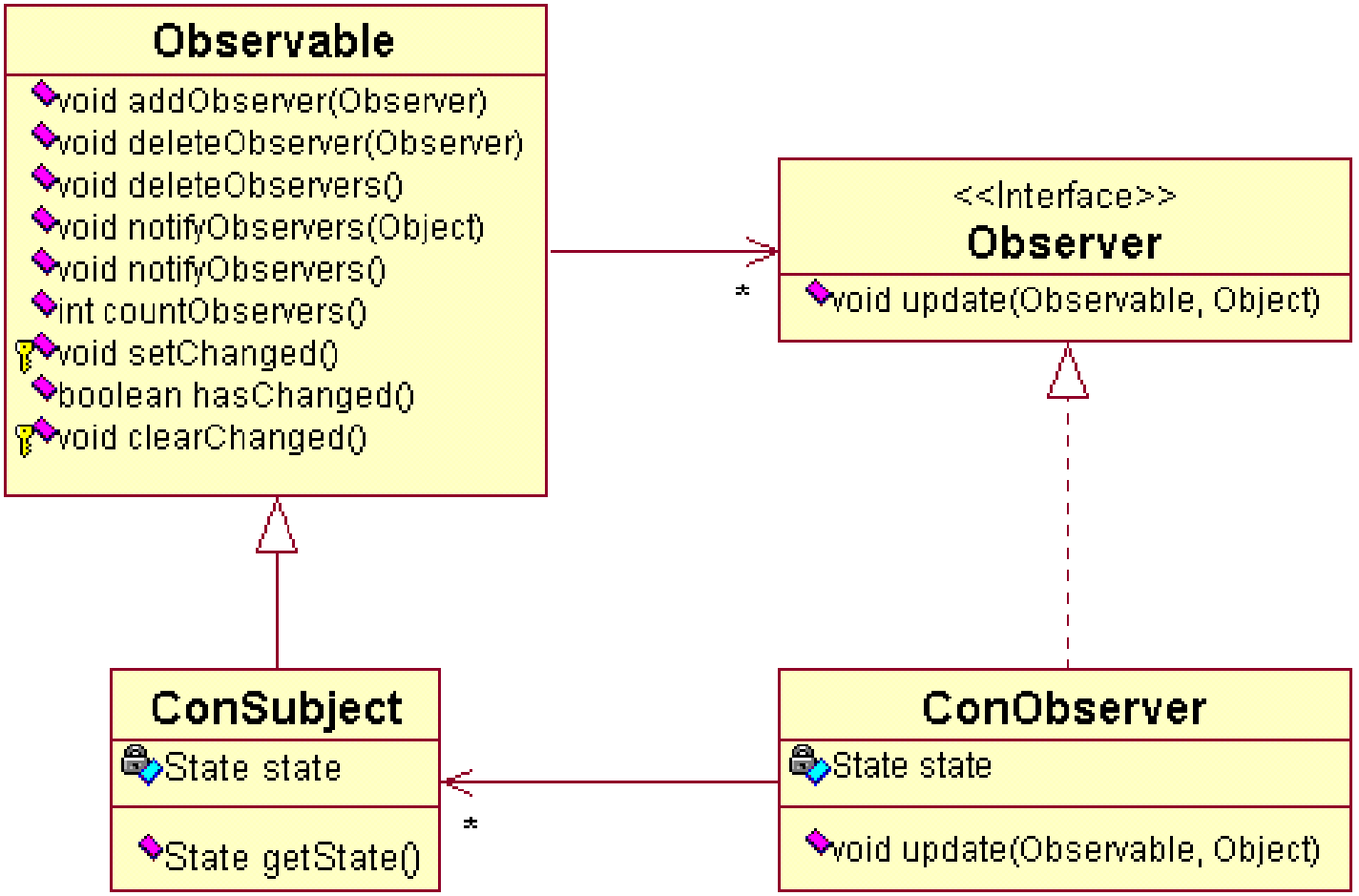
6.2 MVC Java Foundation Classes (JFC)



Bei kleineren Anwendungen wird, wie in den JFC häufig die Benutzungsschnittstelle ((User Interface (**UI**)) gewissermaßen als **Einheit** von **View** und **Controller** gesehen.



6.3 Observable – Observer – Muster (Java)



Observable-Observer ist ein *Pubisher-Subscriber*-Muster

6.4 Bsp:Erweiterung von „mvc1“ auf „mvc2“

In „mvc1“ ist der Dialog DlgConversion nicht modal, d.h. er kann in einer Anwendung mehrfach erzeugt werden.

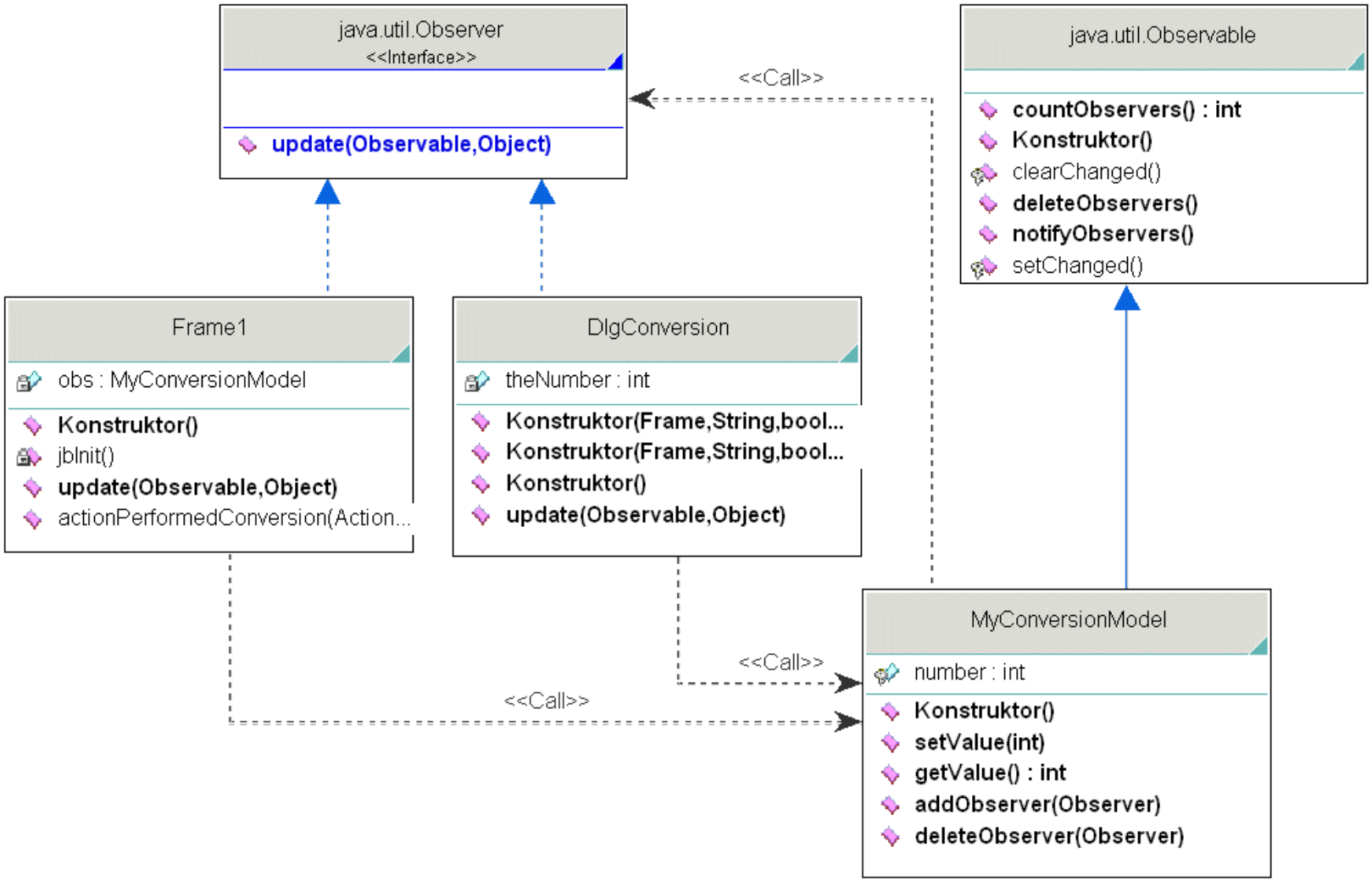
Angenommen

- wir wollen in allen offenen Fenstern die gleiche Zahl darstellen, sobald diese in einem Fenster vom Benutzer geändert wird
- Wir wollen im Haupt-Fenster (Frame1) die Anzahl der offenen Fenster sehen

was müssen wir tun?

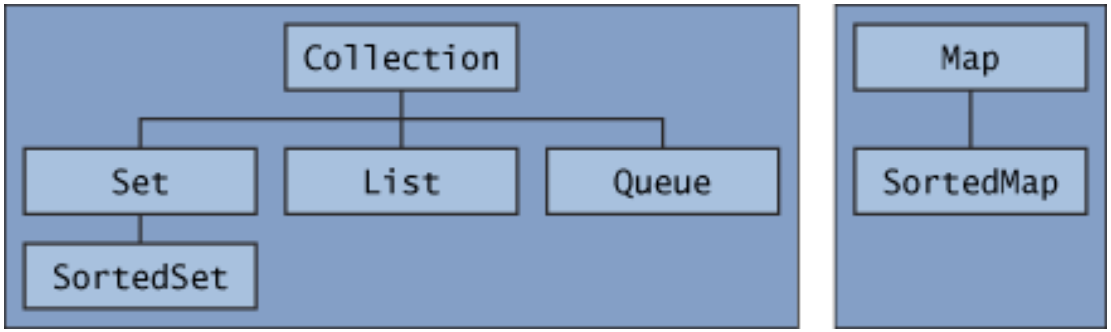
Wie müssen wir die Anwendung erweitern?

6.4 Bsp: Erweiterung auf „mvc2“



6.5 Collections in Java

- **Collections** können allgemein klassifiziert werden in die folgenden Interfaces (siehe auch Literatur)



- **Collections** stellt Gruppe von Objekten dar
- **Set** ist eine Collection, die *keine doppelten* Elemente hat
- **List** ist eine *geordnete* Collection, die doppelte Elemente enthalten kann
- **Queue** stellt ein FIFO dar
- **Map** stellt Paare aus **{ Schlüssel , Objekt }** dar. Schlüssel müssen eindeutig sein.

6.6 Speichern von Objekten in Java

- Neben Datenbanken stellt die **Persistenz** (*etwas dauerhaft machen*) eine elegante Möglichkeit zur Speicherung (**Serialisierung**) von Objekten dar.
- In Java sind alle Objekte, welche das ***Serializable*** Interface implementieren, *serialisierbar*.

```
FileOutputStream fos = new FileOutputStream(fName);
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(object);
```

- Aus gespeicherten (serialisierten) Daten können durch **Deserialisierung** wieder Objekte erzeugt werden.

```
FileInputStream fis = new FileInputStream(fName);
ObjectInputStream ois = new ObjectInputStream(fis);
MyObject object = (MyObject) ois.readObject();
```

- Siehe Dokumentation und Beispiele in der Literatur