Typbindung

bei

Java

Typbindung

- Auffälliger Unterschied zum Pythoncode ist die

Deklaration von Attributen mit ihrem *Typ*

- *"normal"*: Definition im Konstruktor

```
13 public class Stuhl
14 {
15     private int xPosition;
16     private int yPosition;
17     private int orientierung;
18     private String farbe;
19     private boolean istSichtbar;
20     private int breite;
21     private int tiefe;
22
23
24     /**
25      * Erzeuge einen neuen Stuhl mit einer Standardfarbe
26      * an einer Standardposition. (Standardkonstruktor)
27      */
28     public Stuhl() {
29         xPosition = 160;
30         yPosition = 80;
31         farbe = "blau";
32         orientierung = 0;
33         istSichtbar = false;
34         breite = 40;
35         tiefe  = 40;
36     }
```

Typbindung

- Für die Arbeit am Anfang nicht relevant ist die Unterscheidung zwischen
  - statischem Typ und
  - dynamischem Typ
- Ansehen beispielsweise bei
  - Interfaces
  - Sammlungsklassen
    (List, ArrayList, LinkedList, ...)

Typbindung

statischer und dynamischer Typ

- interessant Projekt mit Gruppen (Tischgruppe, Schrankwand, Konferenztisch-Gruppe) *Raumplaner_mit_Gruppen*
- enthält
  `protected ArrayList<Moebel> moebelListe;`
- Konkrete Moebel sind alle Typen, sogar auch Gruppen
- ebenso:
  `public void fuegeHinzu(Moebel moebel)`

Typbindung

Interface Shape

- `protected Shape gibAktuelleFigur() {`

  `…`

  `return <konkretes Shape>; }`

- Bei Stuhl konkret

  `GeneralPath stuhl = new GeneralPath();`

- Bei Tisch konkret

  `Shape tisch =`
  `new Rectangle2D.Double(0,0,breite,tiefe);`

Typbindung

- Es gibt implizite Typumwandlungen.
- Hilfreich bei Ausgaben / Wertzuweisungen von Zahlen als String:

```
int zahl = 3;

String s;

s = "" + zahl ;
```

ist zulässig und ergibt für s die Ausgabe "3".

- Umgekehrt:

```
zahl = new Integer(s).intValue()
```