II.2.4 Konstruktoren

Montag, 30. November 2020 10:11

Erzeugung neuer Objekte:

```
Rechteck r = new Rechteck();
```

Hierbei wird eine Konstruktor-Methode aufgerufen, die ein neues Rechteck-Objekt erzeugt.

Konstruktor:

- Methode zur Erzeugung neuer Objekte
- Heißt genauso wie die Klasse.
- Es wird kein Rückgabetyp angegeben.
- Ziel: Setze die Attribute des neuen Objekts auf bestimmte Werte
- Vorgehen:
  Zunächst erhalten die Attribute die Initialwerte aus der Variablendeklaration.
  Anschließend wird der Code des Konstruktors ausgeführt.

```
class Rechteck {
    int laenge = 5;

    Rechteck () {
        laenge ++;
    }
    ⋮
}
```

Bei `r = new Rechteck();` hätte r.laenge den Wert 6.

Bei r = new Rechteck(); hätte r.laenge den Wert 6.

- Es dürfen mehrere Methoden mit gleichem Namen (und daher auch mehrere Konstruktoren) auftreten, solange ihre formalen Parameter "hinreichend verschieden" sind.
  ⇒ Überladen von Methoden

Überladen

- Verschiedene Methoden dürfen den gleichen Namen haben, falls ihre Parameterlisten "verschieden" sind.
- Parameterlisten gelten als "verschieden", falls:
  - unterschiedliche Anzahl von Parametern oder
  - unterschiedliche Datentypen bei den Parametern
- Unterschiedliche Parameternamen oder unterschiedliche Resultattypen reichen nicht.

```
class A {
    ⋮
    int f(int x) {...}   ← verboten
    ⋮
    double f(int y) {...}
    ⋮
}
```

- Falls beim Aufruf einer überladenen Methode mehrere Implementierungen passen, wird die spezifischste passende Methode ausgeführt.

Überladung ist nur erlaubt, wenn die Auflösung beim [illegible] ist

Überladung ist nur erlaubt, wenn die Auflösung beim Methodenaufruf eindeutig ist.

Bsp: `Rechteck(double l, int b) {...}`
`Rechteck(int l, double b) {...}`

Beim Aufruf von
`new Rechteck(1,2)`
ist unklar, welcher Konstruktor ausgeführt werden soll.
⇒ Abbruch des Programms mit Fehler.

Weiteres Bsp. für überladene Methode:
`System.out.print(...)`
↑
gibt Argument auf Bildschirm aus

⇒ für verschiedene Argumenttypen vorhanden

Wenn man keinen Konstruktor selbst schreibt, dann wird der parameterlose Konstruktor automatisch erzeugt.

Achtung: Sobald man einen Konstruktor implementiert (ggf. mit Parametern), wird in der Klasse kein Konstruktor automatisch erzeugt.

Zugriff auf das "aktuelle" Objekt: "this"

d.h.: auf das gerade erzeugte Objekt im Konstruktor bzw. auf das Objekt, für das eine nicht-statische Eigenschaft berechnet wird

```
class Rechteck {
    :
    double flaeche() {
        return this.laenge * this.breite;
    }
    :
}
```

Aufruf mit `r.flaeche()`

↑ aktuelles Objekt

Weiterer typischer Konstruktor: Kopier-Konstruktor