II.2.4 Konstruktoren

Freitag, 10. November 2017 09:00

Erzeugung von neuen Objekten: `new Rechteck();`

Hierbei wird ein Konstruktor ausgeführt, der die Attribute des neuen Objekts geeignet initialisiert.

Konstruktoren können selbst geschrieben werden:

- Konstruktor heißt genauso wie die Klasse.
- Erzeugt das neue Objekt d. Klasse ⇒ kein Rückgabetyp muss angegeben werden.
- Typischerweise enthält der Rumpf d. Konstruktors Anweisungen zur Belegung der Attribute des neu erzeugten Objekts.
- Konstruktoren können Eingabeparameter haben.
- Es kann mehrere Konstruktoren (gleichzeitig) geben, die verschiedene Arten v. Eingabeparametern haben (Überladung).
- Aufruf v. Konstruktoren mit Hilfe v. "new".
- Vorgehen bei Ausführung d. Konstruktors:
  erst werden Attribute auf Initialwerte aus der Deklaration gesetzt, danach wird Code d. Konstruktors ausgeführt:

```
class Rechteck {
    int laenge = 5; ....
    Rechteck () {
        laenge ++;
        :
    }
    :
}
```

Bei

`r = new Rechteck();`

hätte

r.laenge den Wert 6

}

Überladung v. Methoden

- Verschiedene Methoden dürfen gleichen Namen haben, falls ihre Parameterlisten "verschieden" sind.
- Parameterlisten gelten als "verschieden" falls:
  - unterschiedl. Anzahl von Parametern oder
  - unterschiedl. Datentypen bei den Parametern

z.B. `Rechteck(int x){ ... }` und `Rechteck(double x){ ... }`

- Es reicht nicht, wenn Parameter unterschiedl. Namen haben oder wenn sich nur der Resultat-Typ unterscheidet.

```
int f(int x){ ... }
```
nicht zusammen mit
```
int f(int y){ ... }
```

```
int f(int x){ ... }
```
nicht zus. mit
```
double f(int x){ ... }
```

- Bei Überladung wird immer die speziellste passende Methode ausgeführt.

Bsp:

Bsp:

- new Rechteck() führt Rechteck() aus
- new Rechteck(1,2) führt Rechteck(double l, double b) aus

  Rechteck(int... a) passt auch, aber bei Überladung werden nur dann vararg-Methoden verwendet, wenn keine andere Methode passt.

  Empfehlung: bei Überladung keine varargs benutzen.
- new Rechteck(3.0) führt Rechteck(double Kantenlaenge) aus
- new Rechteck(3) führt Rechteck(int s) aus.

  int ist spezieller als double

  (denn es gibt implizite Typkonversion von int nach double).

  Es wird immer die speziellste passende Methode ausgeführt.

  Bsp: Rechteck(double l, int b) ist spezieller als Rechteck(double l, double b)

  Was passiert bei Rechteck(double l, int b) und Rechteck(int l, double b)

  und Aufruf new Rechteck(1,2)?

  ⇒ Fehler
- new Rechteck(1,2,3) führt Rechteck(int... a) aus

Man kann beliebige Methoden überladen, z.B. System.out.print:

Je nach Typ des Arguments wird unterschiedliche Methoden-implementierung ausgeführt, die Argument in String konvertiert und ausgibt.

Wenn man selbst gar keinen Konstruktor schreibt, dann erzeugt Java automatisch einen parameterlosen Konstruktor.
Achtung: Wenn man selbst Konstruktoren schreibt (z.B. Rechteck(double l)), dann gibt es Rechteck() nicht, es sei denn, man schreibt ihn selbst.

Zugriff auf das Objekt, für das eine nicht-statische Methode/Attribut aufgerufen wird: this

```
class Rechteck {
    ⋮

    double flaeche() {
        return this.laenge * this.breite;
    }
    ⋮
}
```

beim Aufruf r.flaeche() bezeichnet this dann das Objekt r

Bei Konstruktoren bezeichnet "this" das gerade erzeugte Objekt.

Weiterer typischer Konstruktor: "Kopier-Konstruktor"!