Java-OOP-Konstruktor

Zur Renovierung deiner Wohnung brauchst du die Wand- und Deckenflächen der einzelnen Räume, um ausreichend Farbe einzukaufen. Hierzu wird eine Klasse *Raum* erstellt, die die wichtigen Informationen und Maße enthält und daraus die notwendigen Flächen bestimmt.

Attribute:

- name → Name/Bezeichnung des Raums (z.B. Wohnzimmer, Küche,…)
- laenge → Länge des Raums
- breite → Breite des Raums
- hoehe → Höhe des Raums
- anzFenster → Anzahl Fenster im Raum (Es wird ein Standardmaß von 1m x 1,2m angenommen
- anzTueren → Anzahl Türen im Raum (Es wird ein Standardmaß von 0,9m x 2,1m angenommen

Methoden

- Notwendige get- und set-Methoden
- berFlaecheDecke → Berechnet die Fläche der Decke (Entspricht der Grundfläche)
- berFlaecheWaende → Berechnet die Flächensumme aller Wände
- berFlaecheMalen → Gibt die Gesamtfläche aus, also Wände plus Decke abzgl. Türen und Fenster

Aufgabe 1

a) Erstelle die zugehörige Klasse mit Hilfe eines UML-Diagramms, wähle dabei geeignete Datentypen und Zugriffsmodifizierer (private, protected, package, public). Tipp: Die Methoden zur Flächenberechnung erhalten keine Eingabeparameter, liefern aber natürlich die Fläche als Wert zurück.

b) Erstelle einen Raum „wohnzimmer“ mit den Maßen 4,5m x 8m x 2,5m, drei Fenstern und zwei Türen und gib die zu bemalende Fläche aus.

- Ergebnis Deckenfläche: 36m²
- Ergebnis Wandflächen: 45m²
- Ergebnis Malerfläche: 73,62m²

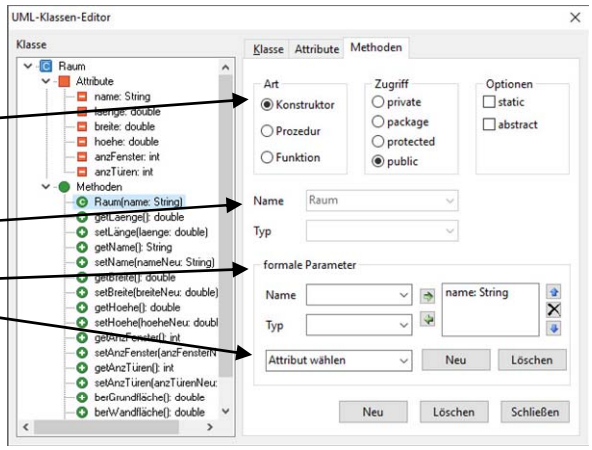
Konstruktor

Wünschenswert ist häufig, dass man nicht über mehrere Zeilen hinweg die einzelnen Attribute mit Werten füllen muss, und die Werte der Attribute sind schließlich das, was die Objekte nachher voneinander unterscheidet, sondern dass die wesentlichen Attribute bereits beim instanziieren des Objekts mit Daten gefüllt werden.

Um gleich beim instanziieren Attribute definieren zu können, wird ein Konstruktor eingesetzt. Es können auch mehrere Konstruktoren mit verschiedenen Signaturen definiert werden, der Konstruktor ist dann „überladen“.

Auch hier ist der UML-Editor behilfliche. Zum Anlegen eines Konstruktors wird eine neue Methode definiert und bei „Art“ dann „Konstruktor“ ausgewählt. Da ein Konstruktor immer den gleichen Namen wie die Klasse, für die er definiert wird, haben muss, wird der Name hier auch gleich fest vergeben.

Unter „formale Parameter“ kann dann direkt bei „Attribut wählen“ das Attribut, bzw. die Attribute, ausgewählt werden, die vom Konstruktor mit Werten besetzt werden sollen.



Attribute, die vom Konstruktor selbst nicht gesetzt werden, werden mit Standardwerten belegt.

Aufgerufen wird eine Klasse mit ihrem Konstruktor nun, indem beim instanziieren die entsprechenden Parameter gesetzt werden.

```
public Raum(String name) {
  this.name = name;
  this.laenge = 0;
  this.breite = 0;
  this.hoehe = 0;
  this.anzFenster = 0;
  this.anzTüren = 0;
}
```

```
public class RaumAufruf{
   public static void main(String[] args){
         Raum wohnzimmer = new Raum("Wohnzimmer");

   }
}
```

Aufgabe 2

a) Erstelle einen Konstruktor, der den Raumnamen entgegennimmt.

b) Führe dein Programm aus Aufgabe 1 aus (ohne etwas sonst zu ändern). Es erscheint eine Fehlermeldung. Wieso erscheint die Fehlermeldung und was müsste in der Klasse geändert werden, damit die Fehlermeldung nicht mehr erscheint.

c) Ändere dein Programm aus Aufgabe 1 so ab, dass die Raumbezeichnung über den Konstruktor direkt gesetzt wird. Führe wiederum die Berechnung durch.

d) Erstelle einen **zusätzlichen** Konstruktor, der neben dem Raumnamen auch die übrigen Angaben zu Maßen, Fenstern und Türen direkt entgegennimmt und setzt.

e) Passe dein Programm wiederum zur Nutzung dieses Konstruktors an.

f) Erstelle drei weitere Zimmer in deinem Programm und gibt die Ergebnisse ansprechend aufbereitet aus.

Anmerkung „this.“

Üblicherweise können Methoden und Attribute eines Objekts über dessen Namen angesprochen werden, z.B. wohnzimmer.berFlaecheMalen(). Wenn innerhalb der Klassendefinition nun die eigenen Attribute und Methoden benutzen wollen, fehlt der Name des Objekts. Daher wird hier statt des Objektnamens der Parameter *this* eingesetzt (this.name, this.anzFenster,…).