Einführung in die Programmierung

by André Karge

Übung - Objektorientierte Programmierung

letzte Woche

- Verzweigungen
- Schleifen

diese Woche

- Besprechung Übungsblatt 2
- Klassen & Objekte
- Konstruktor
- Instanziierung von Objekten
- Methodenaufrufe mit Objekten
- Besprechung Übungsblatt 3

Übungsblatt 02

Übungsblatt 02

Aufgabe 1 - Scope

```
public class Scope {
  public static void main(String[] args) {
    int g = 5;                  // 1
    int h;                      // 2
    h = g * 3;                  // 3
    {
      g = 4;                    // 4
      int j = h - 2;            // 5
      int i;                    // 6
      {
        i = 7;                  // 7
        j = j * 3;              // 8
        byte k = 1;             // 9
      }
      g = i;                    // 10
    }
    int l = g - h;              // 11
    g = h;                      // 12
  }
}
```

Übungsblatt 02

Aufgabe 2 - Kontrollstrukturen

Eclipse Code

Übungsblatt 02

Aufgabe 2 - Kontrollstrukturen

Eclipse Code

Aufgabe 3 - Zahlensysteme

Eclipse Code

Klassen & Objekte

Klassen - Insel 3

Klasse

- Grundkonzept der Objektorientierten Programmierung
- definiert neuen Typen (Komplexer Typ)
- beschreibt Funktionsweise von Objekten dieses Types

Klassen - Insel 3

Klasse

- Grundkonzept der Objektorientierten Programmierung
- definiert neuen Typen (Komplexer Typ)
- beschreibt Funktionsweise von Objekten dieses Types

Objekte

- Ist ein Exemplar (Instanz / Ausprägung) einer Klasse
- Beispiel: Hammer, Meißel, Spachtel sind alles Objekte der Klasse Werkzeug

Klassen - Insel 3

Klasse

- Grundkonzept der Objektorientierten Programmierung
- definiert neuen Typen (Komplexer Typ)
- beschreibt Funktionsweise von Objekten dieses Types

Objekte

- Ist ein Exemplar (Instanz / Ausprägung) einer Klasse
- Beispiel: Hammer, Meißel, Spachtel sind alles Objekte der Klasse Werkzeug

Wozu das ganze? - Um komplexe Softwaresysteme besser modellieren, entwerfen und implementieren zu können

Klassen in Java

- schon in der ersten Übung gesehen:

```
/**
 * @author
 *
 */
public class ClassName {      // Klassenname
  /*
   * Attributes               // Attribute: [visibility] [static] [final] type name;
   */

  /*
   * optional: Constructors   // Konstruktor(en): [visibility] ClassName(...) {...}
   */

  /*
   * Methods                  // Methoden: [visibility] [static] return-type name(...) {...}
   */
}
```

Klassen in Java - Attribute - Insel 3.4.4

- Variablen in Klassen = Objektvariablen
- jede Instanz einer Klasse hat seine eigene Ausprägung
- Objektvariablen bilden den *Zustand* des Objekts (Objektattribute)

Klassen in Java - Attribute - Insel 3.4.4

- Variablen in Klassen = Objektvariablen
- jede Instanz einer Klasse hat seine eigene Ausprägung
- Objektvariablen bilden den *Zustand* des Objekts (Objektattribute)

Beispiel:

```
public class Werkzeug {
  int gewicht;
  String besitzer = "Peter Zwegat";
  ...
}
```

Übungsaufgabe

1. Schreiben Sie eine Main-Klasse mit einer Main-Methode
2. Erstellen Sie eine zusätzliche Klasse 'Autor' mit den Attributen 'name', 'nachname', 'geboren', 'webseite', 'notiz'

Klassen in Java - Instanziierung - Insel 3.4

- nehmen wir an, wir haben eine Klasse definiert
- um ein neues Objekt der Klasse anzulegen nutzt man das Stichwort **new**
- Schema: Klasse bezeichner = new Klasse();

Klassen in Java - Instanziierung - Insel 3.4

- nehmen wir an, wir haben eine Klasse definiert
- um ein neues Objekt der Klasse anzulegen nutzt man das Stichwort **new**
- Schema: Klasse bezeichner = new Klasse();

Beispiel:

```
public class Werkzeug {...} // anlegen einer neuen Klasse

public class Main {
  public static void main(String[] args) {
    Werkzeug hammer = new Werkzeug(); // hammer ist eine neue Instanz der Klasse Werkzeug
  }
}
```

Klassen in Java - Klassenmethoden

- neben Attributen, können Klassen auch Methoden haben
- werden innerhalb einer Klasse angelegt und definieren Funktionen von allen Ausprägungen der Klasse
- Beispiel: eine Klasse 'Auto' hat die Funktion 'fahren', was bedeutet, dass alle Instanzen der Klasse eine Funktion 'fahren' haben

Klassen in Java - Klassenmethoden

- neben Attributen, können Klassen auch Methoden haben
- werden innerhalb einer Klasse angelegt und definieren Funktionen von allen Ausprägungen der Klasse
- Beispiel: eine Klasse 'Auto' hat die Funktion 'fahren', was bedeutet, dass alle Instanzen der Klasse eine Funktion 'fahren' haben

```
public class Werkzeug {
  int gewicht; // anlegen einer Klassenvariable gewicht
  String name; // anlegen einer Klassenvariable name

  public void print() {
    System.out.println("Hallo, ich bin ein Werkzeug");
  }
}
```

Klassen in Java - Klassenmethoden

Aufruf von Klassenmethoden

- Methoden, die in Klassen angelegt sind können nur auf Objektinstanzen der Klasse aufgerufen werden
- visibility von Methoden und Variablen gibt an, ob man außerhalb der Klasse Zugriff auf diese bekommt
- visibility: *public*, *private*, *protected*

Beispiel:

```
public class Main {
  public static void main(String[] args) {
    Werkzeug hammer = new Werkzeug();

    hammer.print(); // Aufruf der Klassenmethode 'print()' auf dem Objekt
  }
}
```

Klassen in Java - Membervariablen

- Zugriff auf Memberattribute über das Stichwort **this.variablename** (Immer im Bezug auf die Ausprägung der Objekte)
- Ein Objekt vom Typen Wekzeug mit *name=Hammer* und *gewicht=1kg* produziert andere Ausgabe beim Aufruf von *printObjectInfo* als ein Objekt mit *name=Feile*

```
public class Werkzeug {
  int gewicht; // anlegen einer Membervariablen gewicht
  String name; // anlegen einer Membervariablen name

  public void printObjectInfo() {
    System.out.println("Hallo, ich bin ein Werkzeug");
    System.out.println("Und zwar bin ich ein " + this.name); // Aufruf der Membervariable über this
    System.out.println("Ich wiege " + this.gewicht + "kg"); // selbiges hier
  }
}
```

Übungsaufgabe - Fortsetzung

3. Instanziieren Sie mehrere Objekte der Klasse Autor innerhalb der Main-Methode der Main-Klasse
4. Erweitern Sie Ihre Autor-Klasse um eine Methode 'printInformation', die alle Attribute auf der Konsole ausgibt und rufen Sie die Methode auf allen instanziiereten Objekten auf.

Klassen in Java - Klassenkonstruktor - Insel 3.4.6

- Konstruktoren sind besondere Methoden von Klassen
- werden beim Instanziieren (**new** Object) von Objekten aufgerufen
- Hat eine besondere Signatur ohne Rückgabetyp: [visibility] ClassName(...) ...

```
public class Werkzeug {
  int gewicht;
  String name;

  public Werkzeug() { // Default Konstruktor
    this.gewicht = 0; // Default wert für gewicht
    this.name = "";   // Default wert für name
  }
  public Werkzeug(int gewicht, String name) { // Überladener Konstruktor
    this.name = name;       // schreibe gegebenen name in die Membervariable
    this.gewicht = gewicht; // schreibe gegebenes gewicht in die Membervariable
  }
}
```

Klassen in Java - Klassenkonstruktor - Insel 3.4.6

```
public class Main {
  public static void main(String[] args) {
    Werkzeug meinWerkzeug = new Werkzeug(); // Aufruf des Default-Konstruktors
    Werkzeug anderesWerkzeug = new Werkzeug(5, "Hammer"); // Aufruf des überladenen Konstruktors
  }
}
```

Übungsaufgabe - Fortsetzung

5. Erweitern Sie Ihre Klasse 'Autor' um einen Standard Konstruktor und einen überladenen Konstruktor
6. Testen Sie Ihre neue Konstruktoren auf neuen Instanzen
7. Probieren Sie eine Methode mit visibility *private* in einer anderen Klasse aufzurufen

Komplexe Zahlen

Komplexe Zahlen

- besteht aus einem real-Anteil z^{real} oder z^{r} und einem imaginär-Anteil $z^{imaginary}$ oder z^{i}
- $z^{complex} = (z^{r}, z^{i})$

Addition mit Komplexen Zahlen

$$z_1 + z_2 = (z_1^r, z_1^i) + (z_2^r, z_2^i) = (z_1^r + z_2^r, z_1^i + z_2^i)$$

Multiplikation mit Komplexen Zahlen

$$z_1 * z_2 = (z_1^r, z_1^i) * (z_2^r, z_2^i) = (z_1^r * z_2^r - z_1^i * z_2^i, z_1^r * z_2^i + z_1^i * z_2^r)$$

Komplexe Zahlen

Addition Beispiel

$z_1 = (1, 2)$
$z_2 = (3, 3)$
$z_1 + z_2 = (1, 2) + (3, 3) = (1 + 3, 2 + 3) = (4, 5)$

Komplexe Zahlen

Addition Beispiel

$z_1 = (1, 2)$
$z_2 = (3, 3)$
$z_1 + z_2 = (1, 2) + (3, 3) = (1 + 3, 2 + 3) = (4, 5)$

Multiplikation Beispiel

$z_1 * z_2 = (1, 2) * (3, 3) = (1 * 3 - 2 * 3, 1 * 3 + 2 * 3) = (3 - 6, 3 + 6) = (-3, 9)$

Fragen?