# Übungen zur Einführung in die Programmiersprache Java

Universität Regensburg
NWF II - Physik
Dominik Köppl

Wintersemester 2011/12
Blatt 7

---

## 26 Dynamische Erzeugung von Objekten

(a) Legen Sie eine Klasse `Person` an, die das Alter und den Vornamen einer Person abspeichern soll.

- Beide Attribute Alter und Vorname sind `private`
- Schreiben Sie Getter-Methoden, um diese auszulesen
- Zusätzlich hat die Klasse einen vollständigen Konvertierungskonstruktor.

Legen Sie ein dynamisch erzeugtes Array aus Personen in der Main-Routine an.

- Fragen Sie zunächst nach der Anzahl der Personen
- erstellen Sie das Arrays
- Lassen Sie für jede Person den Benutzer alle Daten eingeben.
- Das Programm soll am Schluss jede Person am Bildschirm ausgeben.

(b) Erstellen Sie eine Vergleichsklasse, die das Interface `java.util.Comparator<Person>` implementiert, welche das Alter beider Personen vergleicht. Nun können Sie mit der statischen Methode `Arrays.sort(Person[] a, Comparator<Person> c)` ein Personen-Array sortieren. Geben Sie am Schluss das Array sortiert aus!

## 27 Polymorphie

Erweitern Sie die Klasse `Person` oder schreiben Sie eine neue Klasse `Haustier`, sodass diese dazu geeignet ist, als Elternklasse der neu zu erstellenden Klassen `Student`, `Schüler` und `Professor` bzw. `Hund`, `Katze` und `Maus` zu fungieren. Implementieren Sie eine Zählfunktion, die zählen soll, wieviele Studenten, Schüler oder Professoren in einer Liste des Types `Person`, bzw. wieviele Hunde, Katzen und Mäuse in einer Liste des Types `Haustier` sind. Legen Sie anschließend im Hauptprogramm eine Liste mit verschiedenen Personen bzw. Tieren an und mache eine Volkszählung bzw. Haustierzählung. Dabei soll der Zählfunktion diese Liste als Argument übergeben werden.
**Tipp** Mit Polymorphie kommt man leicht ans Ziel.
**Fleißarbeit** Fügen Sie weitere Personenklassen bzw. Tiereklassen hinzu.

## 28 Ein wenig Mathe

Wir bezeichnen eine Menge $F$ als *Field*, falls es die binären Operationen

- $\cdot : F \times F \to F, (x, y) \mapsto x \cdot y$ für die Multiplikation
- $+ : F \times F \to F, (x, y) \mapsto x + y$ für die Addition
- $- : F \times F \to F, (x, y) \mapsto x - y$ für die Subtraktion
- $/ : F \times F \to F, (x, y) \mapsto \frac{x}{y}$ für die Division

und zusätzlich die Elemente

- $e^+ \in F$ neutrales Element der Addition ($x + e^+ = x$ für alle $x \in F$)
- $e^\cdot \in F$ neutrales Element der Multiplikation ($x \cdot e^\cdot = x$ für alle $x \in F$)

besitzt.

(a) Schreiben Sie ein Interface `Field<F>`, welches diese Operationen als Methoden für einen Typ $F$ deklariert. Hierbei sollte man natürlich auf sinnvolle Methodenbezeichner achten.

(b) Legen Sie eine Klasse `Ganzzahl` an, die die Klasse `Number` erweitert und das `Field<Ganzzahl>` implementiert. Intern wird eine `Integer`-Variable verwaltet, die alle abstrakten `Number`-Methoden füllt. Verwenden Sie das Autoboxing der Klasse `Integer` zur Implementierung.

**Tipp** Für eine Ganzzahl ist $e^+ = 0$, $e^\cdot = 1$.