

# Informatik I – Eprog

## Übungsaufgaben WS06/07

Prof. Harald Gall, Beat Fluri,  
Andreas Jetter , Michael Würsch  
Überarbeitet von Andreas Siegrist, Jonas Zuberbühler

### Übung 4

#### **Abgabe: Woche 47**

##### *Hinweise:*

- *Die seriöse und vollständige Bearbeitung und pünktliche Abgabe der Übung ist Bedingung für das Bestehen der Veranstaltung*
- *Die Abgabe hat an Ihre/n Tutor/in in elektronischer Form (Email, Lösungen als Attachment im .java-/.txt-, .ps- oder .pdf-Format<sup>1</sup>) bis spätestens 12:00 Uhr am Vortag der jeweiligen Übungsstunde zu erfolgen*
- *Tragen Sie bitte zu Beginn jedes Dokumentes Ihren Namen inkl. Matrikelnr. ein.*

---

<sup>1</sup> Mögliches Freewaretool finden Sie unter <http://sourceforge.net/projects/pdfcreator/>.

## Aufgabe 1: Konstruktoren und statische Variablen

### Lernziele

1. Sie können mehrere Konstruktoren in einer Klasse implementieren und diese gezielt aufrufen.
2. Sie kennen den Unterschied zwischen Instanz- und Klassenvariablen.

### Aufgabenstellung

Verwenden Sie für die folgenden Aufgaben die der Übung beigelegte Klasse `Employee.java`. (Hinweis: Die Klasse soll Ihnen die Tipparbeit für die accessor-Methoden ersparen. Erwarten Sie in den Zwischentest keine ähnliche Hilfestellung.)

- a) Fügen Sie der Klasse eine Instanzmethode mit der folgenden Signatur hinzu:

```
1 public void raiseSalary(double byPercent) { ...
```

Die Methode soll das Gehalt des jeweiligen Angestellten um eine übergebene Prozentzahl erhöhen (z.B. `byPercent=10` bewirkt eine Gehaltserhöhung um 10%).

- b) Stellen Sie einen Konstruktor bereit, welcher es erlaubt, die Attribute `fLastName`, `fFirstName`, `fStreet`, `fZip`, `fResidence` und `fDateOfBirth` bereits bei der Objekterzeugung mit Werten zu belegen.
- c) Fügen Sie der Klasse `Employee` eine Klassenvariable `sNumberOfEmployees` vom Typ `int` hinzu und initialisieren sie diese mit 0. Erweitern Sie ihren Konstruktor aus b) nun dahingehend, dass dieser den aktuellen Wert von `sNumberOfEmployees` der Instanzvariablen `fID` zuweist und erstere anschliessend um eins erhöht. So soll gewährleistet werden, dass jedes neue `Employee`-Objekt eine eindeutige Personalnummer erhält.
- d) Schreiben Sie einen weiteren Konstruktor, welcher keine Argumente erwartet. Er soll den in c) erweiterten Konstruktor mit Dummy-Werten aufrufen. So könnte für `fLastName` der Wert "TestNachname", für `fFirstName` "TestVorname", etc. übergeben werden. Danach soll der Konstruktor zudem auf dem Bildschirm ausgeben, dass ein Testobjekt generiert wurde und welche Personalnummer dieses besitzt:

*"Ein Testobjekt mit der Personalnummer 12 wurde erzeugt!"*

- e) Implementieren Sie nun eine Methode `equals()`, welche als Argument ein `Employee`-Objekt erwartet. Die Methode soll die Personalnummern der beiden Angestellten vergleichen. Stimmen diese überein, so soll die Methode wahr zurückliefern, andernfalls falsch.
- f) Schreiben Sie eine Methode `isOlderThan()`, welche als Argument ein `Employee`-Objekt erwartet. Die Methode soll überprüfen, ob das übergebene Objekt jünger ist, als das aktuelle und in diesem Fall wahr zurückliefern. Ist es hingegen älter, so wird falsch zurückgeliefert.  
(Hinweis: Sollte Ihnen der Vergleich der beiden Geburtsdaten Kopfzerbrechen bereiten, so wird sich die API (speziell die Beschreibung der Klasse `java.util.GregorianCalendar`) als hilfreich erweisen.)
- g) Fügen Sie der Klasse nun noch eine Methode `toString()` hinzu, welche eine `String`-Repräsentation der `Employee`-Objekte zurückliefert. Ein solcher `String` könnte dann zum Beispiel den folgenden Inhalt haben:

*Hans Muster  
Rämistrasse 74*

*8001 Zürich*  
*Geboren am: 01.01.1970*  
*Personalnummer: 15*

- h) Erstellen Sie nun einen TestDriver, in dessen `main()`-Methode Sie einige Objekte der Klasse `Employee` erzeugen. Verwenden Sie hierzu sowohl den Konstruktor mit, als auch den ohne Argumente. Rufen Sie die verschiedenen `set()`-Methoden, die `setSalary()`- und `raiseSalary()`-Methoden, sowie die `toString()`-Methoden der erzeugten Objekte auf. Führen Sie anschliessend noch ein paar Vergleiche von Angestellten mittels der `equals()`-Methode, sowie der `isOlderThan()`-Methode durch und geben Sie die Resultate am Bildschirm aus.
- i) Warum erzeugt folgendes Codefragment, wenn die Klasse `Employee` damit ergänzt wird, einen Fehler zur Übersetzungszeit? Begründen Sie ihre Antwort und diskutieren Sie den Sinn der Methode.

```
1 public static GregorianCalendar getDateOfBirthStatic() {  
2     return fDateOfBirth;  
3 }
```

## Aufgabe 2: Konstruktoren, Komposition und Delegation

### Lernziele:

1. Sie können das Konzept der Komposition und Delegation anwenden und dies anhand von Klasseninteraktionen zeigen.

### Aufgabenstellung

- a) Schreiben Sie eine Klasse `Robot`, welche einen Konstruktor besitzen soll, der zwei `String`-Objekte als Argumente erwartet: Einen Namen und eine Modellbezeichnung:

```
1 new Robot("Robbie", "R2-D2");
```

- b) Erweitern Sie die Klasse und speziell den Konstruktor aus a) nun so, dass jedes `Robot`-Objekt bei seiner Konstruktion eine eindeutige Seriennummer erhält.
- c) Fügen Sie der Klasse nun eine Methode `identify()` hinzu, welche einen `String` bestehend aus dem Namen, der Modellbezeichnung und der Seriennummer zurückliefert:

*"Robbie, R2-D2, Nummer 12"*

- d) Schreiben Sie anschliessend eine Klasse `Tool`, welche ein Werkzeug modellieren soll. `Tool` soll einen Konstruktor besitzen, welcher wieder zwei `String`-Objekte als Argumente erwartet. Während der erste Parameter einen Bezeichner (z.B. "einen Schraubenzieher") enthalten soll, wird mit dem zweiten Parameter eine kurze Beschreibung zum Verwendungszweck des Werkzeuges übergeben (z.B. "zieht eine Schraube an").
- e) Fügen Sie `Tool` nun noch zwei accessor-Methoden hinzu, die den Bezeichner (`getName()`), sowie den Verwendungszweck (`getPurpose()`) zurückliefern.
- f) Implementieren Sie einen weiteren Konstruktor in der Klasse `Robot`, welcher den Konstruktor aus b) dahingehend erweitert, dass dieser zusätzlich ein Objekt der Klasse `Tool` als Parameter erwartet. Speichern Sie das `Tool`-Objekt in einer Instanzvariablen. (Versuchen Sie dabei Codeduplizität zu vermeiden.)
- g) Erweitern Sie `Robot` nun um eine Methode `useTool()`. Die Methode soll dann u.A. mit Hilfe der Methoden aus c) und e) einen Text der folgenden Art auf dem Bildschirm ausgeben:

*"Robbie, R2-D2, Nummer 12, zieht eine Schraube an und verwendet hierzu einen Schraubenzieher."*

- h) Erstellen sie die Methode `useTool(Tool tool)`. Diese Methode soll das übergebene `Tool`-Objekt in der Instanzvariablen speichern und danach das gleiche Verhalten wie die Methode `useTool()` zeigen.
- i) Erstellen Sie nun einen `TestDriver` und erzeugen Sie in der `main()`-Methode vier `Robot`-Objekte und einige `Tool`-Objekte. Rufen Sie dann die `useTool()`-Methode der `Robot`-Objekte auf.

## Aufgabe 3: Zufallszahl

### Lernziele

1. Sie können mit Hilfe der Klasse `Random` (Pseudo-)Zufallszahlen generieren.
2. Sie wissen, wie man über die Konsole Benutzereingaben einliest und den Programmablauf damit steuert.
3. Sie kennen die Syntax der `while`-Schleife und können diese implementieren.

### Aufgabenstellung

- a) Entwerfen und Implementieren Sie ein Programm, welches mit Hilfe der Klasse `java.util.Random` (die Methoden der Klasse, sowie Beispiele zu deren Benutzung entnehmen Sie bitte der API) eine Zufallszahl `rnd` zwischen 0 und 9 erzeugt. Der Benutzer soll die Möglichkeit erhalten, die Zahl zu erraten und einzugeben. Anschliessend soll das Programm die Eingabe mit der generierten Zahl vergleichen und Folgendes ausgeben:

- "Leider daneben!", falls `|rnd - input| > 1`
- "Knapp daneben!", falls `|rnd - input| == 1`
- "Richtig! Glückwunsch!", falls `rnd == input`

Wenn der Benutzer anstatt einer Zahl "quit" eingibt, soll das Programm mit einer entsprechenden Meldung vorzeitig beendet werden können.