

Prof. Dr. Jürgen Giesl  
Carsten Fuhs, Peter Schneider-Kamp, Stephan Swiderski

## Übung *Programmierung WS 2007/08* - Blatt 5

Die Übungsblätter sollen in Gruppen von je 2 Studierenden bearbeitet werden. Diese 2 Studierenden müssen aus derselben Übungsgruppe kommen. Lösungen müssen bis zum 27. November, 15 Uhr, in den Kasten für Ihre Übungsgruppe eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55).

Bitte vergessen Sie nicht, die **Nummer Ihrer Übungsgruppe** sowie die **Namen und Matrikelnummern** der Mitglieder Ihrer 2er-Gruppe auf **jedes Blatt** Ihrer Abgabe zu schreiben.

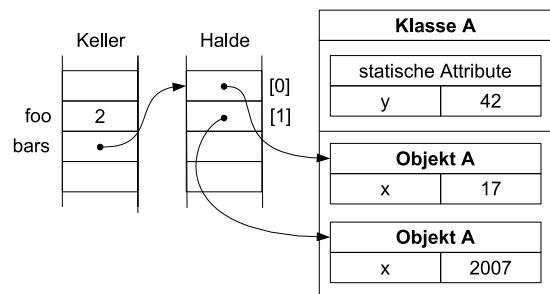
Alle erstellten Java-Programme sind sowohl in gedruckter Form abzugeben als **auch per E-Mail** an den jeweiligen Tutor zu senden.

Neben den Ausgabe-Methoden `System.out.print...` dürfen nur die Bibliotheks-Methoden verwendet werden, die in den für Aufgabe 2 zur Verfügung gestellten Klassen vorhanden sind.

Beachten Sie beim Programmieren unbedingt die in der Globalübung vorgestellten Programmierkonventionen, die auch im WWW unter Globalübung zu finden sind.

### Aufgabe 1 (6 Punkte)

Visualisieren Sie den Speicherzustand an den markierten Stellen beim Aufruf der `main`-Methode der folgenden Klasse (Zeile 31 und 37). Verwenden Sie dabei die in der Vorlesung vorgestellte Notation (siehe Vorlesung Kapitel II.1.5, Folien 7-9, Kapitel II.2.1, Folien 3,7-9 und Kapitel II.2.2, Folien 15-20). Sie brauchen keine Methoden darzustellen. Um statische Attribute zu visualisieren, zeichnen Sie einfach um alle Objekte einer gemeinsamen Klasse einen umgebenden Kasten, wie in der nebenstehenden Abbildung, die einen möglichen Speicherzustand darstellt. Dort und nur dort sollen Sie dann die statischen Attribute der entsprechenden Klasse aufführen.



```

1 public class A {
2
3     public int x = 5;
4     public static int y = 1;
5
6     public A copy() {
7         A res = new A();

```

```
8         res.x = x;
9         return res;
10    }
11
12    public static void subtract(A a, int x) {
13        a.x -= x;
14        x -= 3;
15        ++y;
16    }
17
18    public static void main(String[] args) {
19
20        int x = 7;
21
22        A[] as = new A[4];
23        as[0] = new A();
24        as[0].x = x;
25        as[1] = as[0].copy();
26        as[2] = as[0];
27        x = 8;
28        as[3] = new A();
29        as[3].x = x;
30
31        // Speicherzustand
32
33        for (int i = 0; i < as.length; ++i) {
34            A.subtract(as[i], x);
35        }
36
37        // Speicherzustand
38
39    }
40
41 }
```

## Aufgabe 2 (14 Punkte)

In dieser Aufgabe soll ein (stark vereinfachtes) System zur Verwaltung von Kunden und Konten in einem Geldinstitut umgesetzt werden. Hierbei soll die Bank mehrere Kunden haben können, die wiederum kein, ein, oder mehrere Konten bei der Bank haben können. Sowohl für die Anzahl der Kunden als auch für die Anzahl der Konten (insgesamt für alle Kunden zusammen) hat das System jeweils eine maximale Kapazität.

Nach dem Start des Systems und jeweils nach Ausführung einer Aktion wird ein Menü der folgenden Form angezeigt:

```
=== Bank Hauptmenue ===  
1 : Alle Kunden anzeigen  
2 : Alle Konten anzeigen  
  
3 : Kunden anlegen  
4 : Konto anlegen  
  
5 : Einzahlen  
6 : Abheben  
7 : Ueberweisen  
  
8 : Beenden
```

Bitte waehlen Sie einen Menuepunkt!

Beachten Sie folgende Anforderungen an das System:

- Nach Wahl von Menüpunkt 1 sollen jeweils für alle Kunden Name, Geburtsdatum und Adresse ausgegeben werden.
- Nach Wahl von Menüpunkt 2 sollen jeweils für alle Konten Kontonummer, Inhaber und Guthaben ausgegeben werden.
- Ein Kunde wird eindeutig durch seinen Namen und durch sein Geburtsdatum identifiziert. Weiterhin verfügt das Geldinstitut auch über seine Adresse.
- Ein Konto wird eindeutig durch seine Kontonummer identifiziert. Jedes Konto hat jeweils einen Inhaber und einen Guthabenstand.
- Der Einfachheit halber genügt es, den Kontostand in Euro als nicht-negativen **int**-Wert zu repräsentieren (d.h., es werden nur ganze Euros, nicht aber Eurocents betrachtet, und die Overflow-Problematik kann hier auch außer Acht gelassen werden).
- Beim Abheben von Geld und beim Überweisen soll der Vorgang nur dann ausgeführt werden, wenn das Konto, von dem das Geld abgebucht wird, eine entsprechende Deckung aufweist. Man kann Konten also nicht überziehen.

- Weiterhin genügt es, wenn Sie die Überweisungsfunktionalität nur zwischen Konten bei ein- und derselben Bank implementieren. Sie sollten jedoch bei einer Überweisung sicherstellen, dass sowohl das Konto des Überweisenden als auch das des Empfängers bei der Bank existieren müssen.
- Negative Summen kann man weder abheben, noch einzahlen, noch überweisen.
- Falls eine Aktion nicht ausgeführt werden kann, z.B. weil nicht genügend Geld auf dem Konto ist oder ein zu bearbeitender Kunde nicht im System gefunden werden kann, soll das System eine entsprechende Fehlermeldung ausgeben.

Sie brauchen hier nicht alles selber zu programmieren, sondern sollen auf ein bereits erstelltes Programmgerüst zurückgreifen, welches Sie bei den Materialien zu dieser Übung von der Webseite der Vorlesung herunterladen können. Dieses besteht aus den Dateien `IO.java` (für die Eingabe), `Kunde.java` (welche einen Kunden darstellt), `Konto.java` (welche ein Konto darstellt) und `Bank.java` (welche eine Bank darstellt sowie die Benutzerschnittstelle zur Verfügung stellt). In den Dateien `Konto.java` und `Bank.java` sind einige Stellen mit `// TO DO` markiert. An diesen Stellen sollen Sie Ihre eigene Implementierung einfügen. Modifizieren Sie keine anderen Stellen im Code, und legen Sie auch keine weiteren Klassen bzw. Dateien an. Sie brauchen Ihrem Tutor nur die Dateien zu schicken, die Sie modifiziert haben, und Sie sollten auch nur diese Dateien ausdrucken.

Bedenken Sie bei der Konzeption Ihrer Lösung, welche der Attribute und Methoden in den von Ihnen erstellten Klassen statisch sein sollten. Hier dürfen (noch) alle Methoden und Attribute **public** sein.