Prof. Dr. V. Linnemann
Stefan Werner
Universität zu Lübeck
Institut für Informationssysteme

Lübeck, den 5. April 2012

Algorithmen und Datenstrukturen

Sommersemester 2012

1. Übungsblatt

Aufgabe 1: Java - Primitive Datentypen und Referenzen

Gegeben sei der folgende Java-Code:

```
1  public class Ueb01Aufg1 {
2
3      public static void main(String args[]){
4          int a = 1;
5          int[] b = new int[2]; b[0]=2 ; b[1] = 3;
6          int[] c = new int[2]; c[0]=4 ; c[1] = 5;
7          doSomething(a, b, c);
8          System.out.println(a);
9          System.out.println(b[0] + " " + b[1]);
10         System.out.println(c[0] + " " + c[1]);
11     }
12
13     public static void doSomething(int x, int[] y, int[] z){
14         z = new int[2];
15         x = 6;
16         int[] k = y;
17         k[0] = 7; k[1] = 8;
18         z[0] = 9; z[1] = 0;
19     }
20 }
```

Welches Ergebnis liefert die Ausführung der `main`-Methode? Begründen Sie Ihre Antwort durch ein geeignetes Diagramm, ähnlich wie in der Vorlesung in Abschnitt 0.4. Geben Sie dabei vor allem die Belegung der Variablen unmittelbar nach dem Eintritt bzw. vor dem Austritt der Methode `doSomething(...)` an! (5 Punkte)

Aufgabe 2: Java - einfache Methoden

Immer wieder führt die Berechnung des Schaltjahres zu kuriosen Problemen in der IT-Landschaft [1,2] . Ein Jahr ist dann ein Schaltjahr, wenn es durch 4, aber nicht durch 100 ohne Rest teilbar ist. Sollte es jedoch durch 400 teilbar sein, so handelt es sich ebenfalls um ein Schaltjahr.

Implementieren Sie eine Methode

```
public static boolean istSchaltjahr(int jahr)
```

die nach dem gegebenen Schema überprüft, ob das übergebene Jahr ein Schaltjahr ist. Ist dies der Fall, so soll die Methode `true` zurückliefern, andernfalls `false`. Zur Restwertermittlung dient in Java der Modulo-Operator `%`.

Testen Sie Ihr Programm mit den Jahren 1900, 2000, 2008, 2012, 2100 durch Ausgaben in einer geeigneten `main`-Methode. (5 Punkte)

Aufgabe 3: Java - Objektorientierte Modellierung und statische Klassenvariablen

In dieser Aufgabe soll noch einmal die einfache objektorientierte Modellierung und die Bedeutung von statischen Attributen wiederholt werden.

a) Implementieren Sie eine Klasse `Kunde` mit den privaten Attributen `vorname` und `nachname` vom Typ `String`. Implementieren Sie einen geeigneten Konstruktor sowie die Zugriffsmethoden `gibVorname()` und `gibNachname()`, welche den entsprechenden Wert zurückliefern. Überschreiben Sie zusätzlich die `toString()`-Methode (implizit vererbt durch `Object`), so dass ein `String` bestehend aus dem Vor- und Nachnamen des Kunden zurückgegeben wird. (2 Punkte)

b) Implementieren Sie eine weitere Klasse `Konto`. Modellieren Sie darin die Kontonummer, die letzte vergebene Kontonummer, den Kontostand, den aktuellen Sparzinssatz und den Inhaber vom Typ `Kunde`. Überlegen Sie sich geeignete Datentypen, welche Größen statisch und welche dynamisch zu deklarieren sind, sowie gegebenenfalls als konstant. Initialisieren Sie statische und/oder konstante Größen geeignet. Geben Sie einen Konstruktor an, welcher den Inhaber sowie einen initialen Kontostand erwartet. Die Kontonummer soll dabei impliziert generiert werden, so dass nicht mehrere Konten mit identischer Kontonummer erzeugt werden können. (4 Punkte)

c) Implementieren Sie die Methoden `aendereZinssatz(double zinssatz)`, `gibKontoStand()`, `einzahlen(double betrag)`, `abheben(double betrag)` und `zahleZinsen()`. Welche Methoden sind statisch zu deklarieren und welche nicht? Begründen Sie kurz Ihre Wahl.

   Prüfen Sie die Korrektheit Ihrer Implementierung in einer `main`-Methode, in der Sie einige Kunden und Konten anlegen sowie die verschiedenen Methoden geeignet testen. (4 Punkte)

[1] http://www.golem.de/news/softwarefehler-schaltjahr-ueberforderte-windows-azure-1203-90165.html
[2] http://www.heise.de/newsticker/meldung/Navigation-TomTom-loest-GPS-Problem-1501168.html

Hinweise

- Diese Hinweise gelten für dieses und alle folgenden Übungszettel dieser Veranstaltung; auch wenn diese Hinweise nicht noch einmal explizit angegeben sind.
- Lösungen für die Übungsaufgaben sind (in der Regel) zu zweit abzugeben. Die erste Seite **muss** oben rechts den Namen und Matrikelnummer der Abgebenden sowie die Gruppennummer enthalten. Lösungen ohne Gruppennummer werden mit 0 Punkten bewertet.
- Personen, die eine Lösung zusammen abgeben, müssen der **gleichen** Gruppe angehören. Lösungen mit mehreren Gruppennummern werden mit 0 Punkten bewertet.
- Die Einteilung der Übungsgruppen erfolgt über die Anmeldung unter

  http://www.informatik-vor-2009.uni-luebeck.de/veranstaltungen/

  Dieser Link ist auch über die Übungsseite der Vorlesung erreichbar. Beachten Sie die Einteilung der Übungsgruppen nach Studiengang.
- Zertifikatskriterium: Für das Zertifikat sind 50% der erreichbaren Punkte bei den Übungsaufgaben, einmaliges Vorrechnen in der eigenen Übungsgruppe und das Bestehen der Klausur am Ende des Semesters erforderlich. Die Benotung des Zertifikats wird ausschließlich durch die Note in der Klausur bestimmt.
- **Ausnahmslos** sind Lösungen zu Programmieraufgaben **zusätzlich** per Email einzureichen. Papierabgaben zu Programmieraufgaben ohne E-Mail-Einreichung bzw. E-Mails ohne Papierabgabe werden nicht gewertet. Senden Sie Ihre Quelldateien hierzu an die E-Mail-Adresse *aud**?**@ifis.uni-luebeck.de* wobei Sie das ***?*** durch Ihre Gruppennummer ersetzen. Fügen Sie dem Betreff Ihrer Abgabe-Mail unbedingt die Gruppennummer und die Nummer des Übungsblattes hinzu.
- Alle Abgaben zu Programmieraufgaben müssen eine ausführbare Klasse und somit eine Methode `public static void main(String[] args)` enthalten, welche die zu implementierenden Funktionen ausgiebig testet. Nichtkompilierbare bzw. nichtausführbare Abgaben führen zu erheblichem Punktabzug.
- Kommentieren Sie Ihren Quellcode ausführlich, damit Ihr Tutor Ihre guten, aber vielleicht nicht ganz korrekt umgesetzten Ideen honorieren kann.

Abgabetermin: Donnerstag, den 12. April bis 10 Uhr im Institut für Informationssysteme, 2. OG im Informatik-Neubau (Raum 2.071, Küche neben Sekretariat)