Einführung in die Programmierung

WS 2019/2020, Blatt 02

Bauhaus-Universität Weimar

Prof. Norbert Siegmund
André Karge

Ausgabe: 11.11.2019
Abgabetermin: Montag, 18.11.2019, 11:00
Besprechung: 21.11.2019

Bitte lesen Sie die folgenden Informationen zum Übungsablauf **sorgfältig** durch.

Grundsätzlich – wenn nicht anders angegeben – sind die Lösungen zu den Übungen zu Einführung in die Programmierung jeden **Montag bis spätestens 11:00 Uhr** an André Karge per E-Mail zu schicken.

Schreiben Sie bitte im Betreff Ihrer E-Mail Ihre **Teamnummer** sowie die Nummer des Übungsblattes. In der E-Mail schreiben Sie bitte zusätzlich Ihren **Namen** und **Matrikelnummer**. Die Lösungen für Sie bitte als Java Dateien als Anlage hinzu. Es werden **keine** kompilierten Dateien, wie *.class oder *.jar angenommen.

Übungen müssen von **minimal zwei** und **maximal drei** Studierenden aus derselben Übungsgruppe in einem festen Team bearbeitet werden (Ausnahmen nur auf Anfrage beim Übungsleiter). Pro Team soll die Lösung nur einmal abgegeben werden. Aufgaben sollen **im Team gelöst** und nicht nur vom Team abgegeben werden. Sie müssen mindestens **50%** dieser Punkte für eine Zulassung zur Prüfung erreichen. Das **Abschreiben** identischer Lösungen wird jeweils mit 0 Punkten bewertet.

Bei Fragen oder Unklarheiten wenden Sie sich bitte **vor der Abgabe** des Übungsblattes an den Übungsleiter (per E-Mail oder persönlich). Es soll nie jemand sagen müssen: „Wir haben die Aufgabe nicht verstanden und konnten sie daher nicht bearbeiten."
Weitere Informationen, wie aktuelle Ankündigungen, finden Sie online (`https://www.uni-weimar.de/de/medien/professuren/intelligente-softwaresysteme/lehre/`) unter Einführung in die Programmierung

Aufgabe 1 Schleifen (4 Punkte)

Schreiben Sie ein Java Programm, das eine Zeichenpyramide ausgibt. Ihr Programm soll beim Programmaufruf per Parameter die Anzahl an auszugebenden Zeilen aufnehmen und das zu verwendenede Zeichen c. Die Zeichen sollen zentriert in der jeweiligen Zeile stehen und in jeder Zeile sollen immer gleichviele Zeichen (Leerzeile und c) vorhanden sein.
Die $i$te Zeile enthält $2 * i$ Zeichen c.
Zur Ausgabe der Zeile implementieren Sie die Methode `public static void printChars(int numberOfChars, char toPrint)`, welches das Zeichen `toPrint` `numberOfLines` mal ausgibt. Verwenden Sie diese Methode sowohl um Leerzeichen als auch Zeichenfolgen von c zu schreiben.

Aufgabe 2 Scope (3 Punkte)

Gegeben sei folgendes Programm in Java.

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

- Geben Sie zu jeder Variablen den Bindungsbereich (d.h., in welchen Zeilen sie *deklariert* ist und somit einen Wert erhalten könnte) und den Gültigkeitsbereich (d.h., in welchen Zeilen sie *initialisiert* ist und somit einen Wert hat) an.
- Geben Sie zu jeder Programmzeile die Werte der Variablen an, die auf der linken Seite eines `=` stehen (d.h., die Variablen, denen ein Wert zugewiesen wird).
- Hinweis: Versuchen Sie zunächst, die Aufgabe ohne die Unterstützung von Eclipse (auf Papier) zu lösen!

Aufgabe 3 Kontrollstrukturen (5 Punkte)

Erstellen Sie ein Java-Programm, das für eine Schulnote ihre Bedeutung ausgibt. In verschiedenen Ländern haben die gleichen Noten unterschiedliche Bedeutungen. Betrachten Sie die Notensysteme in Deutschland, den Niederlanden und Spanien. Der Benutzer soll eine nummerische Note (ganze Zahl von 0 bis 10) und Kennzeichen eines Landes (”D” für Deutschland, ”N” für die Niederlande und ”S” für Spanien) eingeben. Abhängig vom gewählten Land, sollen die folgenden Bedeutungen für die Noten ausgegeben werden:

- Deutschland:
  - 1 – sehr gut
  - 2 – gut
  - 3 – befriedigend
  - 4 – ausreichend
  - 5 – mangelhaft
  - 6 – ungenügend
- Niederlande:
  - 10 – ausgezeichnet
  - 9 – sehr gut
  - 8 – gut
  - 7 – ausreichend
  - 6 – genügend
  - 5 – fast genügend
  - 4 – ungenügend
  - 3 – stark ungenügend
  - 2 – schlecht
  - 1 – sehr schlecht
- Spanien:
  - 9- 10 – ausgezeichnet
  - 7- 8 – bemerkenswert
  - 6 – gut
  - 5 – ausreichend
  - 3- 4 – ungenügend
  - 0- 2 – sehr mangelhaft

Wenn eine ungültige Note für ein Land eingegeben wird (z.B. 7 für Deutschland), so soll das Programm eine entsprechende Fehlermeldung an den Benutzer ausgeben.

Hinweis:

- Zwei Strings `s1` und `s2` können auf Gleichheit getestet werden, indem Sie die Methode `equals` der Klasse `String` wie folgt benutzen:

  `s1.equals(s2)` ergibt als Ergebnis `true`, wenn `s1` und `s2` gleich sind, und sonst `false`.

Aufgabe 4 Klassen (2+2+2+3+1 Punkte)

In der Vorlesung haben Sie die Klassen `Point` und `Line` kennengelernt. Wir wollen hier nun zweidimensionale Dreiecke auf verschiedene Weise als Klasse modellieren. Geben Sie die Klassen jeweils in Java Notation an.

(a) Entwerfen Sie eine Klasse `Triangle` mit Attributen, die die Eckpunkte des Dreiecks speichern. Geben Sie eine Konstruktormethode an, die ein solches Dreieck aus 3 Point Objekten erzeugt.

(b) Entwerfen Sie nun eine Klasse `Triangle2`, die in ihren Attributen nicht die Eckpunkte, sondern die 3 begrenzenden Linien des Dreiecks speichert. Geben Sie eine Konstruktormethode an, die ein solches Dreieck aus den 3 begrenzenden Linien erzeugen kann sowie eine, die das Dreieck aus den 3 Eckpunkten erzeugt.

(c) In (a) und (b) kann man statt `Point` und `Line` Objekten auch direkt `float` Werte (für die entsprechenden Koordinaten) speichern. Welche Vor- und Nachteile sehen Sie bei den beiden Ansätzen?

(d) Schreiben Sie ein kurzes Java Fragment auf, welches aus den Punkten `p1`, `p2`, `p3` ein `Triangle` und ein `Triangle2` Objekt erzeugt und anschließend beide Dreiecke um `a` in x-Richtung und `b` in y-Richtung (wobei `a` und `b` als `float a = ...;` und `float b = ...;` gegeben seien) verschiebt, indem die Koordinaten der Punkte bzw. Linien entsprechend geändert werden.

(e) Was müsste an den Klassen `Point`, `Line`, `Triangle` bzw. `Triangle2` verändert werden, wenn statt im zweidimensionalen Raum Dreiecke in drei Dimensionen modelliert werden sollen?