Einführung in die Programmierung

WS 2018/2019, Blatt 02

Bauhaus-Universität Weimar

Prof. Dr.-Ing. Norbert Siegmund
M.Sc. André Karge

Ausgabe: 05.11.2018
Abgabetermin: Montag, 12.11.2018, 11:00 Uhr

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 Komplexe Zahlen (8 Punkte)

Implementieren Sie eine Klasse `Complex`, um komplexe Zahlen darzustellen. Eine komplexe Zahl $z = (a; b)$ besteht aus zwei reellen Zahlen a und b. Auf den komplexen Zahlen kann man Addition und Multiplikation durchführen, die wie folgt definiert sind:

- Addition: $(a; b) + (c; d) = (a + c; b + d)$
- Multiplikation: $(a; b) * (c; d) = (a * c - b * d; a * d + b * c)$

(a) Legen Sie eine weitere Klasse `ComplexTest` an, die mit Hilfe von `Complex`-Objekten folgende Ausdrücke berechnet:

(i) $(1; 2) + (3; 4)$
(ii) $(12.3; 1.01) + (23; 42)$
(iii) $(1; 2) * (3; 4)$
(iv) $(12.3; 1.01) * (23; 42)$

(b) Erweitern Sie Ihre Klasse `Complex` um eine Methode `public boolean equals(Complex z)`, die zwei komplexe Zahlen miteinander vegleicht. Testen Sie Ihr Programm mit den Paaren

- $(1; 2)$ und $(1; 2)$
- $(1; 2)$ und $(3; 4)$

Geben Sie die Ergebnisse der Vergleiche aus.

(c) Testen Sie den folgenden Code und erklären Sie die Ausgabe.

```
Complex z1 = new Complex(1, 2);
Complex z2 = z1;
Complex z3 = new Complex(1, 2);
System.out.println("z1 == z2 -> " + (z1 == z2));
System.out.println("z1 == z3 -> " + (z1 == z3));
```

Aufgabe 2 Geld, Geld, Geld (6 Punkte)

Erstellen sie eine Klasse ”Konto”. In der Klasse sollen mindestens eine Person(komplexer Datentyp) und ein Kontostand repräsentiert werden. Neben einem Konstruktor sollen ebenfalls Methoden implementiert werden, um Geld einzuzahlen oder abzuheben. Beachten sie hierbei, dass der Kontostand nicht unter 0 fallen darf! Testen sie die Funktionen in einer entsprechenden Main Methode.

Aufgabe 3 OOP (6 Punkte)

Deklarieren Sie in Java eine Klasse Student. Geben Sie dabei 4 sinnvolle Attribute an (inkl. Datentypen). Versuchen Sie, Variablen und Konstanten zu verwenden. Von den 4 Attributen sollten wenigstens 2 Attribute unterschiedliche, komplexe Datentypen sein, die ebenfalls definiert werden müssen.
Geben Sie 2 Beispiele für Objekte an, die durch Ihre Klassendeklaration erstellt werden können. Geben Sie 2 Beispiele für Objekte an, die **nicht** durch Ihre Klassendeklaration erstellt werden können.