Einführung in die Programmierung

WS 2018/2019, Blatt 08

Bauhaus-Universität Weimar

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

Ausgabe: 10.12.2018
Abgabetermin: Montag, 17.12.2018, 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 Vererbung (10 Punkte)

Bankkonten gibt es in unterschiedlichen Varianten:

- Standardkonten mit Dispositionskredit, aber ohne Zinsen
- Jugendkonten ohne Dispositionskredit, ohne Zinsen, aber mit einem Abhebungslimit je Monat
- Zinskonten ohne Dispositionskredit, aber mit Verzinsung

Allen Konten gemeinsam sind die Attribute Kontoinhaber, Kontonummer und Kontostand.
Die Zinssätze und das Abhebungslimit können sich im Laufe der Zeit ändern, sehen Sie deshalb Methoden vor, um diese zu setzen. Die Zinsen sollen der Einfachheit halber stets am Monatsende gutgeschrieben bzw. abgezogen werden.

(a) Entwerfen Sie Klassen für die genannten Kontomodelle. Sehen Sie in den Klassen geeignete Attribute und Methoden vor. Nutzen Sie Vererbung aus, um die Konten geschickt zu modellieren. Die Methoden für Ein- und Auszahlungen sollen den Tag des laufenden Monats (alle Monate sollen 30 Tage haben) als Parameter bekommen, damit die Zinsen jeweils anteilig mitgerechnet werden können. Sehen Sie eine Methode vor, die am Monatsende aufgerufen wird und den Rechnungsabschluss (Zinsen berechnen und gutschreiben bzw. abziehen) vornimmt.

(b) Implementieren Sie die Methoden für Ein- und Auszahlungen und zum Setzen der Zinssätze. Achten Sie darauf, die Bedingungen bei Auszahlungen (Dispositionskredit bzw. monatliches Limit) zu berücksichtigen und ggf. die Auszahlung zu verweigern.

(c) Schreiben Sie ein eine Klasse `TestAccounts` mit einer Main-Methode, die Ihre Klassen und Methoden testet.

Aufgabe 2 Generics (6 Punkte)

(a) Erstellen Sie eine Klasse `Pair<A, B>`, die zwei Objekte generischen Typs speichern kann. Versehen Sie diese mit folgenden Methoden:

(i) Einem Konstruktor, mit dem beide Attribute gesetzt werden können.
(ii) Jeweils einen Getter für die beiden gespeicherten Attribute.
(iii) Jeweils einen Setter für die beiden gespeicherten Attribute.
(iv) Eine `toString()` Methode, die die `toString()` Methode der beiden Attribute aufruft und klammert.

(b) Schreiben Sie ein Programm, das einen Textstring analysiert.

Schreiben Sie dazu eine Methode, die als Parameter einen `String` erhält und als Rückgabe ein `Pair<Character, Integer>` liefert, in dem sowohl das am häufigsten im String vorkommende char, als auch die Zahl der Vorkommnisse gespeichert wird. Leerzeichen sollen hierbei nicht beachtet werden. Enthält der String keine Zeichen oder wird eine `null`-Referenz übergeben, so soll `null` zurück gegeben werden. Gibt es mehr als ein Zeichen, das maximal oft vorkommt, so kann ein beliebiges maximal oft vorkommendes Zeichen zurück gegeben werden.

Geben Sie mit Hilfe dieser Methode fest programmierte Strings aus. Testen Sie ihr Programm mit folgenden Strings:

- "Many had seen it as clinching proof that the whole of known creation had finally gone bananas."
- " "
- "ignorance is bliss"

Hinweis: Sie können hierzu die bereits bekannte `String`-Methode `toCharArray()` benutzen.

Aufgabe 3 Interfaces (8 Punkte)

Schreiben Sie einen modularen Zeichenketten-Manipulator. Downloaden Sie sich zuerst das dazu notwendige und auf der Webseite der Professur erhältliche Interface `StringManipulator`.

(a) Implementieren Sie eine Klasse, deren Objekte maximal fünf `StringManipulator`-Objekte speichern können. Diese Klasse soll eine Methode haben, mit der sich diese Objekte hinzufügen lassen. Eine weitere Methode soll auf einem übergebenen `String` nacheinander (in der Reihenfolge, in der die Objekte hinzugefügt wurden) die Methode `modify` der konkreten `StringManipulator`-Implementierungen aufrufen und das Ergebnis zurückgeben.

(b) Erstellen Sie folgende konkrete Implementierungen des Interfaces bzw. dessen Methode `modify`:

- Alle Großbuchstaben des Strings werden in Kleinbuchstaben umgewandelt.
- An den übergebenen String wird ein fester (aber im Konstruktor des Manipulators setzbarer) String anhängt.
- Die übergebenen Strings werden fortlaufend durchnummeriert.

(c) Testen Sie Ihre Implementierung ausführlich.