Einführung in die Programmierung

WS 2018/2019, Blatt 09

Bauhaus-Universität Weimar

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

Ausgabe: 14.01.2019
Abgabetermin: Montag, 21.01.2019, 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 Wrapper (7 Punkte)

(a) Implementieren Sie in Java die Wrapper Klasse für die ganzen Zahlen mit folgenden Methoden:

(i) `public Integer(int value)` erstellt ein `Integer` Objekt, das den Wert `value` repräsentiert.

(ii) `public int intValue()` gibt den Wert des `Integer` Objekts zurück.

(iii) `public static Integer valueOf(int value)` erstellt das `Integer` Objekt, das den `int`-Wert `value` repräsentiert, und gibt es zurück.

(iv) `public void set(int value)` setzt den Wert des `Integer` Objekts auf `value`.

(v) `public boolean equals(Integer other)` vergleicht zwei `Integer` Objekte.

(b) Implementieren Sie eine Testklasse, die die folgenden Methoden definiert:

(i) `public static void incrementInteger(Integer i)` erhöht den Wert des `Integer` Objekts um 1 (analog zu `++` Operator.

(ii) `public static void decrementInteger(Integer i)` verringert den Wert des `Integer` Objekts um 1 (analog zu `--` Operator).

Die oben gegebenen Methoden sollen aus der main Methode der Testklasse aufgerufen werden. In beiden Methoden erfolgt die Übergabe der Parameters nach dem *call by reference* Prinzip.

Aufgabe 2 Exceptions (8 Punkte)

Schreiben Sie eine Klasse mit einer Methode `calcAverage`, die eine `java.util.ArrayList` als Parameter übergeben bekommt. In dieser Liste sollen Schulnoten von 1 bis 6 (nur ganzzahlig) gespeichert sein, deren Durchschnitt von der Methode berechnet und zurückgegeben werden soll. Beachten Sie, dass eine `java.util.ArrayList` keine primitiven Datentypen speichern kann und Sie deshalb eine Wrapper-Klassen (entweder eine eigene oder aus `java.lang`) verwenden müssen.
Die Methode `calcAverage` soll den Durchschnitt nur berechnen und zurückgeben, wenn folgende Bedingungen erfüllt sind:

- die Liste muss mindestens ein Element enthalten
- die Noten dürfen nicht negativ sein
- die Noten müssen zwischen 1 und 6 sein.

Für jeden der Fälle soll eine Exception definiert werden. Dazu legen Sie eigene Exception-Klassen an. Zum Beispiel könnte die Exception für die erste Bedingung folgendermaßen aussehen:

```
public class ListSizeException extends Exception
{
}
```

Ihre Exceptions benötigen also keine weitergehende Implementierung als oben gezeigtes Beispiel. Fangen und behandeln Sie alle Ausnahmen an der Aufrufstelle von `calcAverage`, z. B. in der `main`-Methode. Dabei soll es für jede Exception eine eigene Fehlermeldung geben.

Aufgabe 3 Exceptions (6 Punkte)

Schreiben Sie eine Klasse `StringStorage`. Die Klasse speichert in einem Array Zeichen. Die Länge des Arrays kann per Konstruktor eingestellt werden.
Legen Sie nun eine Klasse `WrongLengthException` an, die von `Exception` abgeleitet ist
(`public class WrongLengthException extends Exception{...}`).
Implementieren Sie in der Klasse `StringStorage` die Methode
`public void storeString(String st) throws WrongLengthException`, die den Parameter `st`, sofern das Zeichenarray genügend Speicherplatz bietet, im Zeichenarray speichert. Ist das Array zu klein um den String zu speichern, so soll die `WrongLengthException` geworfen werden.
Wird die Methode mehrere Male mit verschieden langen Strings aufgerufen, so soll immer nur der letzte String im Zeichenarray gespeichert werden.
Testen Sie nun Ihre Implementierung.