technische universität
dortmund

Fakultät für Informatik
Lehrstuhl 14
Lars Hildebrand

Praktikum zu

Einführung in die Informatik für LogWilngs und WiMas

Wintersemester 2017/18

Übungsblatt 10

Besprechung:
8.–12.01.2018
(KW 2)

Vorbereitende Aufgaben

Aufgabe 10.1: Wiederholung: Klammern

Geben Sie an, wofür folgende Klammern in Java verwendet werden.

- [...]

- (...)

- { ... }

Aufgabe 10.2: Wiederholung: Funktionsparameter

Geben Sie einen geeigneten Methodenkopf für die folgenden öffentlichen, statischen Funktionen an.

- Eine Funktion **average**, die den Durchschnitt eines **double**-Arrays berechnet.

  Beispielantwort: `public static double average(double[] array)`

- Eine Funktion **plus**, die zwei reelle Zahlen miteinander addiert und die Summe zurückgibt.

- Eine Funktion **countWords**, die die Wörter in einem **String** zählt und zurückgibt.

- Eine Funktion **printMaximum**, die das Maximum eines **int**-Arrays mit `System.out.println` auf dem Bildschirm ausgibt.

- Eine Funktion **times**, die einen Integer **n** und einen Integer **x** entgegen nimmt und ein **n** Elemente langes Array, gefüllt mit dem Wert **x** zurückgibt.

Präsenzaufgaben

Aufgabe 10.3: Objektvariablen und -methoden vs. Klassenvariablen und -methoden

In dieser Aufgabe sollen Sie sich mit der unterschiedlichen Verwendung von Objekt- und Klassenelementen vertraut machen.

Manche Zuweisungen und Methodenaufrufe sind im unteren Programm nicht erlaubt (vgl. dazu Folien 39–42 in Kapitel 6). Notieren Sie auf den Linien neben dem Programmtext, ob die jeweilige Zuweisung oder der jeweilige Methodenaufruf erlaubt ist oder nicht.

```
1  class Tester {
2      int var1;
3      static int var2;
4
5      void test1() {
6          var1++;                    _____
7          var2--;                    _____
8      }
9
10     static void test2() {
11         var1++;                    _____
12         var2--;                    _____
13     }
14
15     public static void main(String[] args) {
16         var1 = 1;                  _____
17         var2 = 1;                  _____
18         test1();                   _____
19         test2();                   _____
20
21         Tester testObjekt = new Tester();
22         testObjekt.var1 = 2;       _____
23         testObjekt.var2 = 2;       _____
24         testObjekt.test1();        _____
25         testObjekt.test2();        _____
26
27         Tester.var1 = 3;           _____
28         Tester.var2 = 3;           _____
29         Tester.test1();            _____
30         Tester.test2();            _____
31     }
32 }
```

Aufgabe 10.4: Klassenvariablen Implementierung

- Erweitern Sie die Klassen **Car** und **Vehicle** um eine private Klassenvariable **instanceCounter**, die die Anzahl der erzeugten **Car**- bzw. **Vehicle**-Objekte zählt.
- Geben Sie in Ihrer Testklasse die Anzahl der Instanziierungen aus.
- Sie benötigen hierzu eine funktionierende Lösung der Aufgaben aus Blatt 9.

Aufgabe 10.5: Heapify

In der Vorlesung haben Sie gelernt, wie Sie mit Hilfe der Heap-Datenstruktur eine Menge von Zahlen sortieren können. Dazu war es nötig, die zu sortierenden Zahlen zuerst nacheinander mit der **einsortieren**-Operation in die Datenstruktur hinzuzufügen. Nachdem alle Zahlen einsortiert wurden, können sie durch **Entfernen** der Wurzel und dem anschließenden Wiederherstellen der Heap-Eigenschaft mit der **heapify**-Operation nacheinander in sortierter Reihenfolge entfernt werden. Auf der Veranstaltungswebseite finden Sie eine **unvollständige** objektorientierte Implementierung dieses Verfahrens. Ihre Aufgabe ist es, den Quellcode zu vervollständigen.

Aufgabe 10.6: Private Methoden

In der Implementierung der Heap-Datenstruktur sind nur wenige Methoden als **public** deklariert. Warum sind ein Großteil der Methoden des Heaps **private** und welchen Vorteil bringt das mit sich?

Ergänzende Aufgaben

Aufgabe 10.7: Minimum rekursiv

In einer Aufabe von Blatt 8 haben Sie eine Funktion geschrieben, die das Minimum eines **int**-Arrays findet. Höchstwahrscheinlich haben Sie dies Iterativ mit einer **for**-Schleife gelöst. Diese Funktion wollen wir noch einmal rekursiv implementieren. Legen Sie dazu die Klasse **MinRec** an.

- Schreiben Sie eine Funktion **minArray** mit zwei Parametern: einem Index und einem **int**-Array. Sie soll das Minimum des Arrays ab dem Index zurückgeben.

  Beispiel: Sei das Array `a = {30, 10, 50, 20, 40, 60}`, soll `minArray(2, a)` den Rückgabewert 20 haben.

  Verwenden Sie bei der Implementierung **keine** Schleifen!

- Überladen Sie die Funktion **minArray** mit einer Funktion, die nur ein **int**-Array als Parameter hat. Diese soll das Minimum des **ganzen** Arrays zurückgeben. Rufen Sie dazu die soeben geschriebene **minArray**-Funktion auf.