Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Programmierung 1
WiSe 25/26
Moritz Balg
Ausgabe: 24.11.2025
Bearbeiten bis: 30.11.2025

Übungsblatt 8

- Felder und Referenzen -

Aufgabe 1: Code beurteilen

a) Gegeben ist folgendes Java-Programm:

```
public class Test {
    public static void main(String[] args) {
        double[] a;
        a = new double[1000];
        for (int i = 0; i <= a.length; i++) {
            a[i] = i;
        }
    }
}
```

Was würden Sie sehen, wenn Sie dieses Programm übersetzen? Was würden Sie sehen, wenn Sie dieses Programm ausführen? Begründen Sie ihre Antwort.

Mögliche Lösung:
Übersetzen ohne Probleme. Die Ausführung führt zum Programmabbruch mit einer `ArrayIndexOutOfBoundsException`(siehe spätere Vorlesunugen), weil auf `a[1000]` zugegriffen wird, was ein illegaler Index ist.

b) Gegeben ist folgendes Java-Programm:

```
public class Test {
    public static void main(String[] args) {
        double[] a;
        for (int i = 0; i < a.length; i++) {
            a[i] = a.length - i;
        }
    }
}
```

Was würden Sie sehen, wenn Sie dieses Programm übersetzen? Was würden Sie sehen, wenn Sie dieses Programm ausführen? Begründen Sie ihre Antwort.

Mögliche Lösung:
Übersetzen führt zu einem Fehler, da `a` nicht initialisiert ist (kein Feld angelegt wurde). Ausführen ist damit nicht möglich.

Aufgabe 2: Referenzgleichheit

Gegeben sei ein Feld von Strings. Sie können testweise in ihrem Programm verwenden:

```
String[] feld = {"Hello", "world"};
```

(Zusatzfrage: Was bewirkt diese Deklaration genau? Machen Sie sich dazu eine Skizze.) Entwickeln Sie zwei Methoden, die jeweils ein solches Feld übergeben bekommen und ein doppelt so großes Feld anlegen und als Resultat liefern, in dem

1. die Referenz auf das Element an der i-ten Stelle im übergebenen Feld an der `2 * i`-ten und `2 * i + 1`-ten Stelle im Resultatfeld steht (Referenzen werden kopiert)
2. der Inhalt des Elements an der i-ten Stelle (also ein String) im übergebenen Feld in die `2 * i`-te und `2 * i + 1`-te Stelle im Resultatfeld kopiert (Inhalte werden kopiert).

Hinweis: `"" + s`

Geben Sie weiterhin zwei Methoden an, die jeweils für zwei Felder der Länge `2 * n` für alle `i = 1, ..., n` die beiden Referenzen an der Stelle `2 * i` und `2 * i + 1` überprüfen
1. auf gleiche Referenz
2. ob sie inhaltsgleich sind

Was sehen Sie als Ergebnis, wenn Sie die Resultatfelder aus den beiden ersten Methoden als Argumente verwenden?

Mögliche Lösung:

```
/**
 * String-Felder unterteilsch kopieren.
 *
 * @author Rudolf Berrendorf
 * @version 1.0
 */
public class StringFelderKopieren {
  public static String[] kopiereReferenzen(String[] a) {
    // Ergebnisfeld anlegen
    String[] result = new String[a.length * 2];
    // Referenzen kopieren
    for (int i = 0; i < a.length; i++) {
      result[2 * i] = result[2 * i + 1] = a[i];
    }
    return result;
  }

  public static String[] kopiereInhalte(String[] a) {
    // Ergebnisfeld anlegen
    String[] result = new String[a.length * 2];
    // Inhalte kopieren
    for (int i = 0; i < a.length; i++) {
      result[2 * i] = "" + a[i];
      result[2 * i + 1] = "" + a[i];
    }
    return result;
  }

  public static boolean vergleicheReferenzen(String[] a) {
    // Referenzen vergleichen
    for (int i = 0; i < a.length / 2; i++) {
      if (a[2 * i] == a[2 * i + 1]) {
        return false;
      }
    }
    return true;
  }

  public static boolean vergleicheInhalte(String[] a) {
    // Inhalte vergleichen
    for (int i = 0; i < a.length / 2; i++) {
      if (!a[2 * i].equals(a[2 * i + 1])) {
        return false;
```

```
      }
    }
    return true;
  }

  public static void ausgeben(String ueberschrift, String[] a) {
    System.out.println(ueberschrift);
    for (int i = 0; i < a.length; i++) {
      System.out.println(a[i]);
    }
  }

  public static void main(String[] args) {
    String[] a = {"hallo", "welt"};
    ausgeben("original", a);

    String[] refs = kopiereReferenzen(a);
    ausgeben("refs", refs);

    String[] copies = kopiereInhalte(a);
    ausgeben("copies", copies);

    // Referenzen vergleichen
    System.out.println(vergleicheReferenzen(refs));
    System.out.println(vergleicheReferenzen(copies));

    // Inhalte vergleichen
    System.out.println(vergleicheInhalte(refs));
    System.out.println(vergleicheInhalte(copies));
  }
}
```

Aufgabe 3: Array-Shift

Entwickeln Sie folgende Methode: $\text{arrayShift}(\text{arr}, \text{k}) := \mathbb{Z}^n \times \mathbb{N} \to \mathbb{Z}^n$

Die Methode nimmt ein Array an und schiebt k 0en von links in das Array ein. Die letzten k Elemente fallen hierbei heraus.

Beispiel

```
int[] feld = {1,2,3,4,5,6};
int[] result = arrayShift(feld, 2); // {0,0,1,2,3,4}
```

Mögliche Lösung:

```
  public static int[] arrayShift(int[] arr, int k) {
    int[] res = new int[arr.length];
    for(int i = 0; i < arr.length - k; i++) {
      res[i+k]= arr[i];
    }
    for(int i = 0; i < k; i++) {
      res[i] = 0;
    }
    return res;
  }
```

Aufgabe 4: Klausurnoten 👾

Geben Sie in einer Klasse Klausurnoten eine Methode `public static void verarbeiteKlausurPunktzahl(int[] punkte, int[] punktGrenzen)` an, die Statistiken zu Klausurpunkten erstellt. Der Methode wird ein Feld punkte übergeben, in dem für alle Teilnehmer die erreichten Punkte stehen (also pro Teilnehmer eine Punktzahl). Weiterhin bekommt die Methode ein Feld `punktGrenzen` übergeben, das die Punktobergrenzen für Noten festlegt. Zur Vereinfachung gibt es nur die ganzen Noten 1,2,3,4,5.

Beispiel für die Eingabe:

- punkte = {30, 74, 81, 67, 95, 65}. Die Länge des Feldes entspricht der Anzahl der Teilnehmer an der Klausur, hier also 6. Der erste Teilnehmer hatte 30 Punkte, der zweite 74 usw.
- punktGrenzen = {59, 69, 79, 89}, also von 0 bis einschließlich 59 Punkte ist die Note eine 5, zwischen 60 und 69 eine 4, zwischen 70 und 79 eine 3, zwischen 80 und 89 eine 2 und ab 90 eine 1. Das Feld hat also immer die Länge 4.

Aus diesen Angaben soll die Methode nacheinander in jeweils einer Zeile folgende Informationen auf dem Bildschirm ausgeben:

- Anzahl der Teilnehmer
- die beste und schlechteste Note in der Form x y
- wieviele Studierende bestanden haben (Note 4 und besser), wieviele nicht bestanden haben (in der Form x y)
- Durchschnittspunktzahl
- Ein Histogramm der Noten (wieviele Klausuren eine 1 waren, wieviele eine 2 ...) in der Form Note Anzahl für alle Noten 1-5, jeweils in einer Zeile

Beispiel

Die Ausgabe zum obigen Beispiel würde dann also aussehen:

```
6
1 5
5 1
68.66666666666667
1 1
2 1
3 1
4 2
5 1
```

Aufgabe 5: Referenzen vertauschen 👾

Schreiben Sie in einer Klasse Vertauschen zwei Methoden vertauscheInhalt bzw. vertauscheReferenz, die beide als einziges Argument eine Feld von Strings bekommen und als Ergebnis der Methode jeweils ein **neues** Feld von Strings liefern. Bei beiden Methoden soll der 0-te String des Eingabefeldes an der letzten Stelle des Resultatfeldes stehen, der 1-te String des Eingabefeldes an der vorletzten Stelle, ..., der letzte String des Eingabefeldes an der 0-ten Stelle des Resultatfeldes stehen. Die beiden Methoden unterscheiden sich darin, dass in der Methode vertauscheInhalt für den i-ten String ein neuer, aber inhaltsgleicher String erzeugt wird, und in der Methode vertauscheReferenz die i-te Referenz in das Ergebnisfeld kopiert wird, also der identische String mit der gleichen Referenz wie der Ursprungsstring. Damit Sie diesen Unterschied in der Wirkungsweise der beiden Methoden auch selber schon testen können, geben Sie zwei weitere Methoden an:

- `boolean testInhaltGleich(String[] arg1, String[] arg2)`
- `boolean testReferenzGleich(String[] arg1, String[] arg2)`

die zwei gleich lange Felder darauf testen, ob sie inhaltsgleich sind (der Inhalt der beiden i-ten Strings muss gleich sein, aber nicht notwendigerweise die Referenzen) oder auch zusätzlich Referenz-gleich, also die beiden i-ten Feldelemente auf den gleichen String verweisen.

Selber testen können Sie ihre vier Methoden, indem Sie die übergebenen Kommandozeilenargumente als Testfeld nehmen.

Beispiel

Beispielaufrufe für solche Plausibilitätstests in `main`:

```
String[] inhalt = vertauscheInhalt(args);
String[] referenz = vertauscheReferenz(args);

// Tests:
// beide müssen gleiches Inhalte haben
if (!testInhaltGleich(inhalt, referenz)) {
    System.out.println("Fehler: Inhalte stimmen nicht ueberein!");
}
// aber beide müssen unterschiedliche Referenzen haben
if (testReferenzGleich(inhalt, referenz)) {
    System.out.println("Fehler: Referenzen muessen unterschiedlich sein!");
```

Aufgabe 6: Statistik

Geben Sie drei Methoden an, die für ein Zahlenvektor $x = (x_1, ..., x_n) \in \mathbb{R}^n$ mit $n \geq 0$ die folgenden Werte berechnen (für das harmonische Mittel muss für alle i gelten: $x_i \neq 0$):

1. Arithmetisches Mittel: $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$
2. Harmonisches Mittel: $\bar{x}_{harm} = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$
3. Standardabweichung: $\sigma(x) = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2)}$

Für $n = 0$ liefern alle Methoden den Wert 0. Geben Sie weiterhin eine Methode an, die für beliebig viele Eingabezahlen von der Kommandozeile in einem `String`-Feld, dessen `String`-Inhalte Zahlenwerte darstellen, einen Feld mit `double`-Zahlen anlegt und dieses als Resultat der Methode liefert. Rufen Sie in `main()` die Umwandlungsmethode auf mit dem main-Argument `args`. Rufen Sie anschließend mit dem resultierenden `double`-Feld die drei statistischen Auswertemethoden auf und lassen sich das Ergebnis jeden Aufrufs auf dem Bildschirm ausgeben. Nutzen Sie die Java-Methode `Math.sqrt`, um die Quasratwurzel zu berechnen.

Beispiel

Aufruf: `java Statistik 1 2 3 3 4 5`

Ausgabe:

```
arithmetischer Mittelwert: 3.0
harmonischer Mittelwert: 2.2929936305732483
Standardabweichung: 1.4142135623730951
```

Mögliche Lösung:

```
**
 * Statistische Werte berechnen.
 *
 * @author Rudolf Berrendorf
 * @version 1.0
 */
public class Statistik2 {
  public static double arithmetischerMittelwert(double[] a) {
```

```
    double summe = 0.0;
    for (int i = 0; i < a.length; i++) {
      summe += a[i];
    }
    if (a.length > 0) {
      return summe / a.length;
    } else {
      return 0.0;
    }
  }

  public static double harmonischerMittelwert(double[] a) {
    double summe = 0.0;
    for (int i = 0; i < a.length; i++) {
      // Änderung: a[i] == 0 wäre möglich und wird hier nicht abgefangen
      summe += 1.0 / a[i];
    }
    if (summe != 0) {
      return a.length / summe;
    } else {
      return 0.0;
    }
  }

  public static double standardabweichung(double[] a) {
    // Bestimme arithmetischen Mittelwert
    double mittelwert = arithmetischerMittelwert(a);
    double summe = 0.0;
    for (int i = 0; i < a.length; i++) {
      // Änderung: Summand sollte mit dem Mittelwert subtrahiert werden, nicht addiert
      double summand = a[i] - mittelwert;
      summe += summand * summand;
    }
    if (a.length > 1) {
      return Math.sqrt(summe / (a.length - 1));
    } else {
      return 0.0;
    }
  }

  public static double[] umwandeln(String[] stringWerte) {
    double[] zahlenWerte = new double[stringWerte.length];
    for (int i = 0; i < stringWerte.length; i++) {
      zahlenWerte[i] = Double.parseDouble(stringWerte[i]);
    }
    return zahlenWerte;
  }
  public static void main(String[] args){
    double[] a = umwandeln (args);
    System.out.println("arithmetischer Mittelwert: " + arithmetischerMittelwert(a));
    System.out.println("harmonischerMittelwert: " + harmonischerMittelwert(a));
    System.out.println("Standardabweichung: " + standardabweichung(a));
  }
}
```

Aufgabe 7: Zahlendarstellung 👾

Geben Sie in einer Klasse `Zahlendarstellung` eine Methode `public static int[] ermittleZiffern(int x, int b)` an, die zu einer Zahl $x \geq 0, x \in \mathbb{N}$ und einer Basis b mit $b \in \mathbb{N}, b > 1$ alle Ziffern der Zahl x zur Basis b bestimmt und in einem `int`-Feld als Resultat liefert. Im Resultatfeld stehen an den Stellen mit kleineren Indizes die niederwertigsten Stellen von x, in feld[0] steht also die niederwertigste Ziffer. Ist die Basis b größer als 10, so nehmen Sie als 'Ziffern' jenseits der 9 die Zifferwerte 10, ..., b – 1 (siehe die Ziffer 13 im letzten Beispiel unten). Beachten Sie, dass $x = 0$ möglich ist. Sie dürfen keine String-Operationen verwenden! Nutzen Sie geschickt ganzzahlige Division und Modulo. Betrachten Sie den Fall $x = 0$ als Sonderfall.

Beachten Sie, dass das Feld genauso groß sein soll, wie der Wert an Ziffern benötigt. Dies müssen Sie also für eine Zahl x erst ermitteln.

Beispiel

- `ermittleZiffern(4711, 10)` liefert das Feld `{1,1,7,4}`
- `ermittleZiffern(4711, 16)` liefert das Feld `{7,6,2,1}`
- `ermittleZiffern(4711, 2)` liefert das Feld `{1,1,1,0,0,1,1,0,0,1,0,0,1}`
- `ermittleZiffern(1234, 16)` liefert das Feld `{2,13,4}`

Info

Im Feld werden laut Vorgabe die niederwertigsten Ziffern an der niederwertigsten Positionen im Feld gespeichert. Wenn Sie *testweise* ihr Ergebnisfeld auf dem Bildschirm ausgeben wollen, so macht es Sinn, den Inhalt in umgekehrter Reihenfolge auszugeben, die höchstwertigen Ziffern zuerst, also der normalen Darstellungsweise.

Aufgabe 8: Abstrakte Datentypen(1)

Basierend auf dem Abstrakten Datentyp Folge mit dessen vorgegebenen Funktionen, definieren Sie eine rekursive Funktion $istenthalten : M^* \times M \rightarrow B$, die für eine Folge f und ein Element a der Grundmenge mit $istenthalten(f, a)$ feststellt, ob das Element a in der Folge f enthalten ist. Sie können Elemente der Basismenge M mit dem Operator $=$ direkt vergleichen. Testen Sie Ihre Lösung an folgendem Beispielaufruf: $istenthalten(< a, b, c, d >, c)$. Geben Sie dabei alle Zwischenschritte in der Berechnung des Beispiels an.

Mögliche Lösung:

$$istenthalten(f, a) = \begin{cases} falsch & \text{falls } isemptyfolge(f) \\ wahr & \text{falls } a = first(f) \\ istenthalten(rest(f), a) & \text{sonst} \end{cases}$$

$$\begin{aligned} istenthalten(< a, b, c, d >, c) &= istenthalten(rest(< a, b, c, d >, c)) \\ &= istenthalten(< b, c, d >, c) \\ &= istenthalten(rest(< b, c, d >, c)) \\ &= istenthalten(< c, d >, c) \\ &= wahr \end{aligned}$$