Übungen zu Algorithmen und Datenstrukturen für Medizintechnik
Sommersemester 2021
René Raab, Robert Richer

Lehrstuhl für Maschinelles Lernen und Datenanalytik
Department Artificial Intelligence in Biomedical Engineering (AIBE)
(https://www.mad.tf.fau.de)

Aufgabenblatt 2 vom 25. April 2021, Abgabe am 09. Mai 2021, 22:00 Uhr

---

»Eine neue Programmiersprache lernt man nur, wenn man in ihr Programme schreibt.«
– *Dennis M. Ritchie, Entwickler von Unix und C*

Es reicht daher nicht aus, die Tafel- bzw. Rechnerübung zu besuchen, Sie müssen auch eigenständig zu Hause programmieren üben!

Aufgabe 2.1: Theorie: Kontrollstrukturen

Punkte siehe StudOn
Kontrollstrukturen

Aufgabenstellung und Abgabe (individuell, nicht als Gruppe!) im StudOn.

Aufgabe 2.2: Grundlagenübung Schleifen

11 Punkte
Schleifen

1. Legen Sie ein neues Java-Projekt `02-Loops` mit einer Klasse `Loops` an. Alle in dieser Teilaufgabe folgenden Codeabschnitte sollen in der `main`-Methode dieser Klasse implementiert werden.

2. Implementieren Sie zuerst die folgenden Schleifen. Alle Ausgaben sollen dabei **in einer Zeile** (getrennt durch ein Leerzeichen, also z. B. »`1 2 3 ...`«) auf `stdout` erfolgen. Nachdem eine Schleife alle gewünschten Werte ausgegeben hat, soll natürlich ein Zeilenumbruch erfolgen.

   Wichtig: Geben Sie in Ihrer Abgabe außer der geforderten Zahlenketten **keine anderen Ausgaben** auf `stdout` aus. Ihre Abgabe sollte **exakt 16** Zeilen ausgeben: 6 Zeilen für Aufgabe 2.1.2, je 2 Zeilen für Aufgaben 2.1.3 und 2.1.4 und 6 Zeilen für Aufgabe 2.1.5

   - Erstellen Sie eine `for`-Schleife, die von 0 bis 10 zählt und die Zahlen auf `stdout` ausgibt.
   - Erstellen Sie noch einmal eine Schleife, die die Zahlen von 0 bis 10 ausgibt. Wenn Sie zuvor den `<` Operator als Abbruchbedingung verwendet haben, verwenden Sie dieses Mal `<=`, falls Sie `<=` verwendet haben, verwenden Sie jetzt `<`.
   - Schreiben Sie neue `for`-Schleifen, die jeweils die folgenden Zahlen ausgeben sollen:
     - ... in 6er-Schritten von 6 bis inkl. 42.
     - ... in 2er-Schritten von 23 bis inkl. 11.
     - ... alle *geraden* Zahlen von 8 bis 17.
     - ... in 2er-Potenz-Schritten (1, 2, 4, 8, ...) von 16 bis inkl. 711.

   Achtung: Lösungen, bei denen die Schleifen *alle* Zahlen durchlaufen, anstatt die nur die gewünschten, erhalten 0 Punkte auf die jeweilige Teilaufgabe!

3. Wandeln Sie die folgenden `for`-Schleifen in `while`-Schleifen um! Falls notwendig, dürfen Sie dafür auch neue Variablen anlegen. Die Benutzung von `break` ist in dieser Aufgabe **nicht** erlaubt.

```
int n = 7;

for (int i = 0; i <= n; i++) {
  System.out.print(i + "␣");
}
System.out.println();

for (int i = 42; 2 * i > n; i-=6) {
  System.out.print(i + "␣");
}
System.out.println();
```

4. Schreiben Sie die beiden `for`-Schleifen aus der vorherigen Teilaufgabe als `while`-Schleifen, setzen Sie die Bedingung der Schleife aber nun auf `true` (also `while (true) { ... }`). Verwenden Sie stattdessen im Rumpf der Schleife eine `if`-Abfrage und eine `break`-Anweisung, um die Schleife zu terminieren.

5. **Babylonisches Wurzelziehen:** Mit dem iterativen Heron-Verfahren konnte man (vermutlich) bereits vor über 2000 Jahren die Näherung von Quadratwurzeln $\sqrt{a}$ berechnen. Dazu legt man einen Startwert $x_0 \neq 0$ fest und berechnet dann schrittweise

$$x_{n+1} = \frac{1}{2} \cdot \left(x_n + \frac{a}{x_n} \right)$$

für $n = 0, 1, 2, 3, \ldots, n_{\max}$. Wie Sie vielleicht an den Werten für n schon sehen können, kann man dieses Verfahren besonders gut mit Schleifen umsetzen.

   a) Deklarieren Sie zuerst in der `main`-Methode die Variable `int nMax` (sie entspricht $n_{\max}$ in der Formel) und initialisieren Sie sie mit dem Wert 5. Legen Sie zudem die Variable `x` geeigneten Typs an, die das Ergebnis der Berechnung speichern soll (entspricht x_n in der Formel). Legen Sie außerdem die Variable `a` an, die dem a aus der Formel entspricht. Initialisieren Sie `x` und `a` mit dem Wert 2.

   b) Schreiben Sie den Schleifenkopf der `for`-Schleife, die sich um das Hochzählen von n kümmert: Die Laufvariable sollten Sie als `int n` deklarieren. Setzen Sie die Initialisierung und die Abbruchbedingung der Schleife entsprechend der Beschreibung des Verfahrens.

   c) Kümmern Sie sich nun um den Schleifenkörper:

      i. Berechnen Sie x_{n+1} und speichern Sie das Ergebnis in der Variable `x`. Beachten Sie, dass wir in unserem Programm sowohl x_n als auch x_{n+1} in der Variable `x` speichern und `x` in jedem Berechnungsschritt einfach nur überschrieben werden soll.

      ii. Geben Sie den aktuellen Wert der Näherung in folgendem Format auf `stdout` aus:

      ```
      Approximation at step ⟨n⟩: ⟨x⟩
      ```

      Ersetzen Sie in der Ausgabe die Werte von ***n*** und ***x*** durch die entsprechenden Variablen.

6. Geben Sie die Datei `Loops.java` mit allen programmierten Schleifen ab.

Aufgabe 2.3: Grundlagenübung Felder 14 Punkte

Felder, Ausgabe

1. Legen Sie ein neues Java-Projekt `02-Arrays` an. Laden Sie aus dem StudOn das Material zu dieser Aufgabe herunter und fügen Sie die Datei `Arrays.java` in Ihr Projekt ein. Diese Datei enthält die Klasse `Arrays`, die bereits ein Gerüst für diese Aufgabe vorgibt. Bei der Bearbeitung der folgenden Aufgaben geben wir Ihnen mithilfe dieses Gerüsts vor, in welcher Methode Sie eine bestimmte Funktionalität implementieren müssen.

   Achtung: Nichteinhalten dieser Vorgaben führt zu 0 Punkten auf die jeweilige Teilaufgabe!

   Wichtig: Beachten Sie **bei allen Teilaufgaben**, dass das Programm auch dann korrekt funktionieren muss, wenn andere Arrays als die folgenden beiden benutzt werden! **Auch leere Arrays sind möglich.**

2. **Initialisieren**
   Legen Sie in der `main`-Methode zwei neue Array-Variablen `array1` und `array2` vom Typ `int[]` an und initialisieren Sie sie **direkt** mit den Werten

   a) 0, 5, 12, 17, 21, 42

   b) 23, 18, 3, 16, 41, 7, 9

3. **Ausgabe**
   Wir möchten unsere Arrays nun gerne auf `stdout` ausgeben. Implementieren Sie zu diesem Zweck die Methode `printArray`. Erstellen Sie eine `for`-Schleife, um die Werte aus dem Array `array`, das als Parameter an die Methode übergeben wird, auszugeben. Geben Sie das Array im folgenden Format auf die Standardausgabe aus und schließen Sie die Zeile nach dem Array mit einem Zeilenumbruch ab. Beachten Sie, dass Kommas nur *zwischen* den Werten stehen dürfen! Testen Sie ihre Implementierung von `printArray`, indem Sie in der `main`-Methode `printArray` nacheinander mit den beiden Arrays aufrufen (z.B. `printArray(array1)`). Ihre Ausgabe sollte nun wie folgt aussehen:

   ```
   [0, 5, 12, 17, 21, 42]
   [23, 18, 3, 16, 41, 7, 9]
   ```

   Hinweis: `printArray` könnte beim Testen der folgenden Teilaufgaben nützlich sein.
   Diese Methode, also sein Programm mittels Ausgaben auf der Konsole zu testen und somit nach eventuellen Fehlern zu suchen, nennt man auch »*print debugging*«.

4. **Summe / Mittelwert**
   Berechnen Sie jeweils die Summe und den Mittelwert beider Arrays. Implementieren Sie die Summe in der vorgegebenen Methode `sum` und den Mittelwert in `mean`. Rufen Sie die Methoden wieder in der `main`-Methode auf und geben Sie das Ergebnis für beide Felder in oben gezeigtem Format aus. Die Summe eines leeren Arrays soll 0 sein. Für den Mittelwert dürfen Sie annehmen, dass das Array mindestens ein Element enthält.

   ```
   array1: sum = ⟨sum⟩, mean = ⟨variable⟩
   array2: sum = ⟨sum⟩, mean = ⟨variable⟩
   ```

   Hinweis: Die beiden Methoden müssen den Wert der Berechnung zurückgeben (im Gegensatz zum einfachen Ausgeben auf `stdout`). Geben Sie daher Ihr Ergebnis am Ende der Methode mittels `return <ergebnis>;` zurück.

5. **Vektoraddition (naja fast)**
   Wir möchten nicht nur einzelne Arrays aufsummieren, sondern auch zwei Arrays elementweise addieren können. Implementieren Sie diese Teilaufgabe in der Methode `sumArrays`. Legen Sie in der Methode ein neues Array `array3` an, das später die Summe enthalten soll. Denken Sie daran, dass Array am Ende der Methode per `return array3;` zurückzugeben.

   Die Arrays sollen elementweise addiert werden. Der i-te Eintrag von `array3` ist also die Summe der i-ten Einträge von `array1` und `array2`. Geben Sie das neu erstellte Array genauso wie in Schritt (3) in der `main`-Methode auf der Standardausgabe aus.

   Für die Länge des neuen Arrays soll gelten: $length(\text{array3}) = min(length(\text{array1}), length(\text{array2}))$.

6. **Maximum**
   Bestimmen Sie das größte Element von `array1`! Implementieren Sie zu diesem Zweck die Methode `maximum` und testen Sie sie in der `main`-Methode. Verwenden Sie zur Implementierung eine Schleife und eine Variable, in der Sie den größten bisher gefundenen Wert speichern.
   Der Algorithmus soll für *beliebige* Arrays funktionieren, Sie können aber davon ausgehen, dass das Feld mindestens die Länge 1 hat. Geben Sie das gefundene Maximum dann auf der Standardausgabe aus.

7. **Tail**
   In dieser Teilaufgabe sollen die beiden Arrays um je ein Element gekürzt werden. Implementieren Sie dazu die Methode `tail`. Legen Sie ein neues Array mit identischem Typ und einer Größe, die um 1 geringer ist als das ursprüngliche Array, an.
   Setzen Sie nun den Inhalt des neuen Arrays mithilfe einer Schleife so, dass er dem des alten, aber ohne dessen *erstes* Element, entspricht.
   Geben Sie das kürzere Feld wie folgt auf `stdout` aus:

```
array1: tail = [5, 12, 17, 21, 42]
array2: tail = [18, 3, 16, 41, 7, 9]
```

8. **Sortierung prüfen**
   In dieser Teilaufgabe soll ermittelt werden, ob die Elemente in den beiden Arrays sortiert sind. Implementieren Sie die Methode `checkSorting`. Verwenden Sie dazu eine Schleife, die durch das Array läuft und prüft, ob die Elemente in aufsteigender Reihenfolge im Array eingetragen wurden. Verlassen Sie die Schleife, sobald das Ergebnis feststeht (sortiert/nicht sortiert). Geben Sie das Ergebnis des Tests auf `stdout` aus:

```
array1: sorted = true
array2: sorted = false
```

9. **Gerade Werte ermitteln**
   In dieser Teilaufgabe soll ermittelt werden, welche Einträge von `array1` gerade und welche ungerade sind. Implementieren Sie die Methode `evenNumbers`. Legen Sie ein neues Feld geeigneter Länge und geeigneten Datentyps an. Weisen Sie in dem neuen Feld den Einträgen für jede gerade Zahl `true` und für jede ungerade Zahl `false` zu. Die Fallunterscheidung soll zusammen mit der Zuweisung in **einer Zeile** erfolgen!
   Geben Sie das neu erzeugte Array auf der Standardausgabe aus. Hierfür können Sie nicht die bereits implementierte `printArray`-Methode verwenden, da diese nur für Integer-Arrays funktioniert. Implementieren Sie stattdessen die Methode `printBooleanArray`, um das erzeugte Array auszugeben.

10. Überprüfen Sie noch ein Mal ob Ihre Methoden auch für andere Arrays funktionieren (beachten Sie auch den Hinweis am Anfang der Aufgabe). Geben Sie dann die Datei `Arrays.java` ab!

Sollte Ihr Programm nicht übersetz- bzw. ausführbar sein, wird die Lösung mit 0 Punkten bewertet. Stellen Sie also sicher, dass IntelliJ IDEA keine Fehler in Ihrem Programm anzeigt, Ihr Programm übersetz- und ausführbar ist sowie die in der Aufgabenstellung vorgegebenen Namen und Schnittstellen *exakt* eingehalten werden. Geben Sie am Schluss die Dateien `Loops.java` und `Arrays.java` über die EST-Webseite ab. Wenn Sie die Aufgabe zusammen mit einem Übungspartner bearbeitet haben, geben Sie im EST unbedingt dessen Gruppenabgabe-Code an! Kontrollieren Sie, ob Ihre Namen am Anfang aller Dateien angegeben sind – schreiben Sie im Quellcode Ihre Angaben in einen Kommentar. Im EST-Abgabesystem können Sie modifizierte Dateien mehrfach abgeben. Nur die zuletzt hochgeladene Version wird bewertet.