Prof. Dr. Ralf Hinze
Markus Heinrich, M.Sc.

TU Kaiserslautern
Fachbereich Informatik
AG Programmiersprachen

Übungsblatt 1: Programmierpraktikum 2021

Ausgabe: 27. April 2021
Abgabe: 04. Mai 2021, 15 Uhr

Hinweis zum Template Auf der Website finden Sie ein Template als `.zip` Datei. Darin ist ein vollständiges Projekt für IntelliJ IDEA enthalten. Sie müssen die Datei auf Ihrem Rechner entpacken und dann den Ordner über den Button "Open or Import" öffnen. Wichtig ist dabei, dass Sie den Ordner auswählen, der die `.iml` Projektdatei enthält, nicht etwa einen übergeordneten Ordner oder die `.iml` Datei selbst.

Die zu implementierenden Methoden sind bereits vorgegeben, der Rumpf der Methode wirft eine Ausnahme. Sie müssen die Zeile `throw new RuntimeException("TODO");` durch Ihren Code ersetzen. In noch nicht implementierten Methoden sollten Sie diese Zeile stehen lassen, da sie sicherstellt, dass die Datei zusammen mit den Tests kompiliert. Außerdem ist für jede Aufgabe eine `main` Methode enthalten, die die zu implementierenden Methoden aufruft und Ausgaben auf die Konsole schreibt. Dies ist zur Fehlersuche gedacht. Sie können den Code der `main` Methode nach Belieben verändern.

Ein IntelliJ IDEA Projekt gibt die zu verwendende JDK Version vor. Im bereitgestellten Projekt haben wir das JDK Version 16 ausgewählt. Wenn Sie eine andere Version installiert haben, dann klicken Sie nach dem Importieren des Projekts auf File -> Project Structure und wählen Sie (wenn links Project ausgewählt ist) unter Project SDK eine installierte JDK Version aus. Sie können jedes JDK ab Version 11 verwenden.

Schauen Sie sich bitte die auf unserer Website verlinkten Videos zu IntelliJ IDEA an. Uns ist bewusst, dass die IDE sehr umfangreich ist und viele Funktionen bietet, die Sie für die Übungsblätter noch gar nicht benötigen. Sie sollen sich aber schon jetzt schrittweise mit der IDE vertraut machen, damit Ihnen in wenigen Wochen die Entwicklung der größeren Projekte deutlich leichter fällt.

Die im Projekt enthaltenen Testfälle können Sie benutzen, um Ihre Lösung vor dem Abgeben zu prüfen, ansonsten erfolgt die Prüfung aber auch durch das Exclaim System. Bestandene Testfälle sind keine Garantie für eine richtige Abgabe. Mehr zu JUnit Tests werden wir in den Projekten kennenlernen, hier werden die Dateien lediglich als zusätzlicher Service für Sie mit bereitgestellt.

Hinweis zur Abgabe Für jede Aufgabe enthält das bereitgestellte Projekt eine eigene Datei (`Aufgabe1.java` bis `Aufgabe3.java`, zu finden im `src` Ordner). Bearbeiten Sie die jeweilige Aufgabe nur in der entsprechenden Datei und laden Sie im Exclaim System *nur diese veränderte* Datei bei der jeweiligen Aufgabe hoch.

Eigenständiges Arbeiten / Plagiate Lösen Sie die Aufgaben bitte eigenständig und nur zusammen mit den Ihnen zugewiesenen Abgabepartner*innen. Sie dürfen gerne externe Materialien verwenden, müssen aber in ihrer Lösung die **Quelle** (Angabe des Fachbuchs, des Fachartikels oder der URL) und den **Umfang** benennen (haben Sie nur die Idee übernommen oder gar den kompletten Quelltext). Wenn sie im Internet recherchieren, bedenken Sie bitte, dass viele Seiten von zweifelhafter Qualität sind und die bloße Übernahme von Programmen nicht zum Lernerfolg beiträgt. Sollten Sie Code kopieren, ohne dabei die Quelle anzugeben, handelt es sich um ein **Plagiat**. Sollten Sie Code mit anderen Teams teilen, handelt es sich ebenfalls um ein Plagiat. **Bei Plagiaten werden alle beteiligten Teams vom Programmierpraktikum ausgeschlossen und können den Schein erst im nächsten Jahr erwerben!**

Aufgabe 1 Vektoren und Matrizen (10 Punkte)

In dieser Aufgabe sollen Sie sich mit Arrays vertraut machen. Dazu implementieren wir einige Operationen auf Vektoren und Matrizen, die als Arrays repräsentiert werden.

Verwenden Sie zur Lösung dieser Aufgabe keine Bibliotheksfunktionen.

a) Implementieren Sie die Methode `int[] add(int[] v, int[] w)`, welche die Vektoren `v` und `w` addiert. Sie können davon ausgehen, dass die Vektoren die gleiche Dimension haben. Erzeugen Sie für die Rückgabe ein neues Array (überschreiben Sie `v` und `w` nicht).

Beispiel:

```
int[] v = {1, 2, 3};
int[] w = {4, 5, 6};
add(v, w); // [5, 7, 9]
```

Hinweis: Die Addition zweier Vektoren v und w erfolgt komponentenweise. An der Stelle i im Ergebnisvektor steht also die Summe der i-ten Komponenten $v_i + w_i$.

b) Implementieren Sie die Methode `int ssp(int[] v, int[] w)`, welche das Standardskalarprodukt der Vektoren `v` und `w` berechnet. Sie können davon ausgehen, dass die Vektoren die gleiche Dimension haben. Überschreiben Sie `v` und `w` nicht.

Beispiel:

```
int[] v = {1, 2, 3};
int[] w = {4, 5, 6};
mul(v, w); // 32
```

Hinweis: Das Standardskalarprodukt der Vektoren v und w ist definiert als $v \cdot w := v_1 \cdot w_1 + \ldots + v_n \cdot w_n = \sum_{i=1}^{n} v_i w_i$. Achten Sie darauf, dass Arrays mit dem Index 0 beginnen.

c) Implementieren Sie die Methode `int[] mul(int[][] m, int[] v)`, welche die Matrix `m` mit dem Vektor `v` multipliziert. Sie können davon ausgehen, dass `m` eine $(i \times j)$-Matrix ist (und aus Zeilenvektoren besteht) und `v` die Dimension j hat. Erzeugen Sie für die Rückgabe ein neues Array.

Beispiel:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 14 \\ 32 \\ 50 \\ 68 \end{pmatrix}$$

wird repräsentiert durch

```
int[][] m = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}};
int[] v = {1, 2, 3};
mul(m, v); // [14, 32, 50, 68]
```

Hinweis: Die i-te Zeile des Produkts einer Matrix A mit n Zeilen und m Spalten und einem Vektor v mit m Zeilen ist gegeben durch $\sum_{j=1}^{m} A_{ij} \cdot v_j$.

Aufgabe 2 Magische Sequenzen (8 Punkte)

Wir bezeichnen eine Sequenz aus Booleschen Werten als magisch, wenn sie entweder nur aus dem Wert `false` besteht oder wenn auf den Wert `true` zwei magische Sequenzen folgen. Wir werden im Folgenden eine solche Sequenz als Array mit Booleschen Werten darstellen.

Implementieren Sie die Funktion `boolean magic(boolean[] arr)`, die feststellt, ob ein übergebenes Boolesches Array magisch ist.

Beispiele:

```
magic []                                  // false, also nicht magisch
magic [false]                             // true, also magisch
magic [true]                              // false
magic [true, false, false]                // true
magic [true, false, true, false, false] // true
```

Verwenden Sie zur Lösung dieser Aufgabe keine Bibliotheksfunktionen.

Hinweis: Durchlaufen Sie das Array und merken Sie sich nach jedem Schritt wie viele magische Sequenzen Sie noch erwarten.

Aufgabe 3 Haustiere füttern (freiwillige Zusatzaufgabe)

Die Funktion `String werIsst(String futter)` aus der Klasse `Aufgabe3Utils` modelliert das Fressverhalten der Nagetiere `"Flecki"`, `"Hoppel"` und `"Fred"`. Zu einem Futter, das als String übergeben wird, sagt uns die Funktion welches unserer drei Tiere das Futter besonders gerne mag, z.B.

```
Aufgabe3Utils.werIsst("Blumenkohl") // "Flecki"
Aufgabe3Utils.werIsst("Brokkoli")   // "Fred"
Aufgabe3Utils.werIsst("Drops")      // "Hoppel"
Aufgabe3Utils.werIsst("Fenchel")    // "Hoppel"
Aufgabe3Utils.werIsst("Heu")        // "Flecki"
```

Um die ungeduldigen hungrigen Tiere möglichst schnell füttern zu können, sortieren Sie Ihren Futterkorb, den wir durch ein Array von Strings modellieren. Implementieren Sie dazu die Funktion `void futterSortieren(String[] futterkorb)`, die den Futterkorb so sortiert, dass vorne im Array das Futter von Flecki liegt, in der Mitte das von Fred und am Ende das von Hoppel. Führen Sie die Sortierung *in-place* durch, also direkt auf dem übergebenen Array, ohne ein neues Array zu erzeugen.

Innerhalb der drei Blöcke muss das Futter nicht sortiert werden.

Ihr Algorithmus soll eine Laufzeit linear zur Länge des Arrays haben.

Verwenden Sie zur Lösung dieser Aufgabe keine Bibliotheksfunktionen.

Beispiele:

```
String[] futterkorb = {"Blumenkohl", "Brokkoli", "Drops", "Fenchel", "Heu"};
futterSortieren(futterkorb); // [Blumenkohl, Heu, Brokkoli, Fenchel, Drops]
```