Heinrich Heine Universität Düsseldorf

Programmierung – WS 2025/26
Dr. Markus Brenneis
Abgabefrist für Feedback:
17.11.25, 10 Uhr

5. Übungsblatt (Lösungsvorschlag[1])

Aufgabe 1: BugHunt: Newtonverfahren nach VL 8

M.[2] hat ein Programm `Newton` geschrieben, das die folgende Aufgabe lösen soll. Den Code von M. finden Sie in den vorgegebenen Dateien zum Übungsblatt im Ilias. Da M. das Programm ohne zwischendurch zu testen von oben nach unten runtergeschrieben hat, enthält es leider einige Fehler. Korrigieren Sie die Fehler im vorgegebenen Programm; versuchen Sie, den Originalcode so wenig wie möglich zu ändern.

> Wenn man keinen Computer hat, um Quadratwurzeln zu berechnen, kann man auf Verfahren zurückgreifen, die die Wurzel näherungsweise berechnen. Die Umsetzung eines Verfahrens wollen wir in dieser Aufgabe betrachten.
>
> Mit dem Newtonverfahren[3] kann die Quadratwurzel einer nicht-negativen, reellen Zahl c näherungsweise bestimmt werden. Der Algorithmus funktioniert wie folgt:
>
> 1. Setze t_0 auf c.
> 2. Wiederhole, bis $t_i^2 \approx c$ die gewünschte Genauigkeit hat:
>   - Setze t_{i+1} auf den Mittelwert von t_i und $\frac{c}{t_i}$.
>
> Das letzte t_i ist dann eine Näherung für $\sqrt{c}$.
>
> Das Java-Programm `Newton` bekommt die reelle Zahl c als Argument übergeben, soll die Quadratwurzel näherungsweise mit dem oben beschriebenen Algorithmus berechnen und auf der Standardausgabe ausgeben. Die Näherung soll so gut sein, dass der Betrag der Differenz von t_i^2 und c kleiner als 0,0001 ist. Wenn dem Programm nicht genau ein Argument oder eine negative Zahl c übergeben wird, soll eine Fehlermeldung ausgegeben werden, die mit `ERROR` beginnt.
>
> In dem Code soll keine Math-Funktionalität außer `Math.abs` (insbesondere nicht `Math.sqrt`) verwendet werden.

[1] Bei den meisten Programmieraufgaben gibt es mehr als einen funktierenden Lösungsweg. Diskutieren Sie gerne untereinander Ihre Lösungsansätze und lerne Sie damit verschiedene Lösungsstrategien und verschiedene Anwendungsmöglichkeiten der Java-Funktionalitäten kennen. Ihre Abgabe müssen Sie aber final selbst formulieren/eintippen.

[2] Der vollständige Name ist der Redaktion bekannt.

[3] https://de.wikipedia.org/wiki/Newtonverfahren

Beispielaufrufe:

```
% java Newton -2
ERROR: ungültiges Argument
% java Newton 0
0.0
% java Newton 4
2.0000000929222947
% java Newton 2
1.4142156862745097
```

Aufgabe 2: Minimum & Maximum nach VL 8

Wir messen an verschiedenen Messstation die Temperaturen an verschiedenen Orten. Wir wollen uns von einem Programm berechnen lassen, welche die höchste und welche die niedrigste gemessene Temperatur ist.

Schreiben Sie ein Programm `MinMax`, welches eine beliebige Anzahl ganzer Zahlen als Argumente von der Konsole entgegennimmt und sowohl das Minimum, als auch das Maximum dieser Zahlen als ganze Zahl auf der Standardausgabe ausgibt. Geben Sie in der ersten Zeile das Minimum und in der zweiten Zeile das Maximum aus. Wenn kein Argument übergeben wird, soll eine beliebige Fehlermeldung ausgegeben werden, welche mit `ERROR` beginnt.[4]

Beispiel:

```
> java MinMax.java
ERROR: Mindestens eine Zahl als Argument übergeben.
> java MinMax.java 3
3
3
> java MinMax.java 5 20 -1 4 15
-1
20
```

Aufgabe 3: Minesweeper nach VL 8

Beim Spiel *Minesweeper* sehen Spieler:innen ein rechteckiges Spielfeld mit verdeckten Feldern, hinter denen Minen versteckt sind. Ziel ist es, alle Felder mit Minen zu finden. Als Hinweis wird beim Aufdecken eines Feldes angezeigt, wie viele Minen sich in den 8 benachbarten Feldern befinden („Nachbarschaftszahl“). Folgende Abbildung zeigt exemplarisch ein vollständig aufgedecktes Spielfeld:

[4] Wie eigentlich überall bei der Programmierung müssen Sie bei Ihren Ausgaben auch auf Groß- und Kleinschreibung achten: Geben Sie `ERROR` aus, nicht `error` und auch nicht `Error`. Das genaue Einhalten solcher Vorgaben mag Ihnen bei unseren Übungsaufgaben (zu Recht) sinnlos erscheinen, aber spätestens wenn Sie in Teams arbeiten, ist die präzise Umsetzung von Vorgaben wichtig und darauf trainieren wir Sie hier schon mal.

basierend auf einem Screenshot von KMines

In den vorgegebenen Dateien zu diesem Übungsblatt finden Sie ein fertiges Programm `Minesweeper`, das die Position der Minen übergeben bekommt und diese „Nachbarschaftszahlen“ ausgibt. Für das Beispiel oben funktioniert das Programm wie folgt:

```
$ java Minesweeper.java 6 5 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1
  1 0 1 1 0 1 0
-1 2 0 2 -1 2
-1 3 0 2 -1 2
-1 2 0 2 3 3
2 3 2 3 -1 -1
1 -1 -1 3 -1 3
```

(*Die Zeile* `1 0 1 1 0 1 0` *gehört noch zu den Argumenten; die Argumente stehen eigentlich alle in einer Zeile, das hat aber so nicht auf das Blatt gepasst.*)

Die ersten zwei Argumente geben Breite und Höhe des Spielfeldes an (jeweils nicht negativ). Die folgenden Argumente geben zeilenweise von links nach rechts an, ob sich an der entsprechenden Position eine Mine (1) oder keine Mine (0) befindet. Ausgegeben werden zeilenweise die „Nachbarschaftszahlen“, wobei ein Minenfeld die Zahl -1 bekommt.

```
$ java Minesweeper.java 2 2 1 0 0
ERROR: 4 Felder erwartet, aber 3 angegeben
```

a) Machen Sie sich mit der Funktionsweise des vorgegebenen Codes vertraut. Beantworten Sie dazu an den drei mit `F:` gekennzeichneten Stellen die dort angegebenen Fragen. Schreiben Sie Ihre Antworten als Kommentar in den Quelltext.

b) Passen Sie die Ausgabe des Programms so an, dass statt `0` ein Leerzeichen und statt `-1` ein `x` ausgegeben wird. Geben Sie nach dem Spielfeld außerdem dessen Größe und die Anzahl der Minenfelder nach dem Schema `25 fields, 6 mines` aus.

```
$ java Minesweeper.java 6 5 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 0 0 0
  0 1 1 0 1 1 0 1 0
x 2   2 x 2
x 3   2 x 2
x 2   2 3 3
2 3 2 3 x x
1 x x 3 x 3
30 fields, 10 mines
```

Wie immer: Geben Sie die kommentierte und angepasste java-Datei mit einem Kommentar `// Feedback gewünscht` ab, wenn Sie Feedback erhalten wollen.

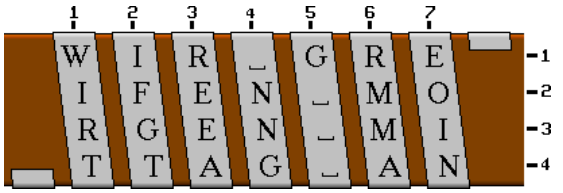
Aufgabe 4: Skytale nach VL 8

„Die Skytale [...] ist das älteste bekannte militärische Verschlüsselungsverfahren. Von den Spartanern wurden bereits vor mehr als 2500 Jahren geheime Botschaften nicht im Klartext übermittelt. Zur Verschlüsselung diente ein (Holz-)Stab mit einem bestimmten Durchmesser (Skytale).

Um eine Nachricht zu verfassen, wickelte der Absender ein Pergamentband oder einen Streifen Leder wendelförmig um die Skytale, schrieb die Botschaft längs des Stabs auf das Band und wickelte es dann ab. Das Band ohne den Stab wird dem Empfänger überbracht. Fällt das Band in die falschen Hände, so kann die Nachricht nicht gelesen werden, da die Buchstaben scheinbar willkürlich auf dem Band angeordnet sind. Der richtige Empfänger des Bandes konnte die Botschaft mit einer identischen Skytale (einem Stab mit dem gleichen Durchmesser) lesen. Der Durchmesser des Stabes ist somit der geheime Schlüssel bei diesem Verschlüsselungsverfahren.“[5]

Nehmen Sie an, Sie können auf dem mit Pergamentband umwickelten Stab n Buchstaben herum und m Buchstaben nebeneinander schreiben. Dabei sei $n = 4$ und $m = 7$. Beispiel:

Klartext: `WIR GREIFEN MORGEN MITTAG AN`




```
       1   2   3   4   5   6   7   -> m
 _____________________________________
|n    |   |   |   |   |   |   |   |   |
1     | W | I | R |   | G | R | E |   |
2  ___| I | F | E | N |   | M | O |___|
3 |   | R | G | E | N |   | M | I |
4 |   | T | T | A | G |   | A | N |
  |___|___|___|___|___|___|___|___|_______
```

Geheimtext: `WIRTIFGTREEA NNGG   RMMAEOIN`

Weil wir weder Pergament noch Stab besitzen, wollen wir ein Programm `Skytale` schreiben, das für uns die Skytale-Verschlüsselung umsetzt. Dieses soll sich wie folgt verhalten:

- Das Programm soll mit zwei Argumenten aufgerufen werden können. Das erste Argument ist entweder `E` oder `D`. Das zweite Argument ist die geheime Nachricht.
- Wird das Programm mit `E` aufgerufen, wird auf der Standardausgabe die verschlüsselte Nachricht ausgegeben.
- Beim Aufruf mit Argument `D` wird die entschlüsselte Nachricht auf der Standardausgabe ausgegeben.
- Wenn die Nachricht nicht genau $4 \times 7 = 28$ Zeichen hat, zu wenige Argumente angegeben sind oder nicht `E` oder `D` als erstes Argument übergeben wurde, soll eine Fehlermeldung, die `ERROR` enthält, ausgegeben werden.

Beispiele:

```
> java Skytale.java E "WIR GREIFEN MORGEN MITTAG AN"
WIRTIFGTREEA NNGG   RMMAEOIN
> java Skytale.java D "WIRTIFGTREEA NNGG   RMMAEOIN"
WIR GREIFEN MORGEN MITTAG AN
> java Skytale.java E "WIR GREIFEN HEUTE UND MORGEN MITTAG AN"
ERROR
> java Skytale.java X "WIR GREIFEN MORGEN MITTAG AN"
ERROR
```

[5] Seite „Skytale“. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 26. Januar 2021, 13:14 UTC. URL: https://de.wikipedia.org/w/index.php?title=Skytale&oldid=208095148

Tipps:

- Stellen Sie zunächst sicher, dass Sie verstanden haben, wie das Verfahren funktioniert, bevor Sie sich Gedanken ums Programmieren machen. Probieren Sie das Verfahren erstmal selbst an einem Beispiel aus.
- Da es sich hier um eine sehr umfangreiche Aufgabe handelt, sollten Sie zunächst überlegen, wie Sie das Gesamtproblem in geeignete Teilprobleme zerschlagen können und wie Sie Ihre Teillösungen testen können. Versuchen Sie gerne zusammen mit Kommiliton:innen einen Programmablauf auf Papier zu entfernen.
- Wenn Sie keine Idee haben, finden Sie auf dieser Seite ein paar Hinweise.