Programmieren in Java

http://proglang.informatik.uni-freiburg.de/teaching/java/2017/

line-numbers

Zeilennummerierung

Woche 03 Aufgabe 1/4

Herausgabe: 2017-05-08
Abgabe: 2017-05-19

Achtung: beachten Sie unbedingt die allgemeinen Hinweise zur Abgabe auf der Homepage.

Project `line-numbers`
Package `linenumbers`
Klassen

`Main`
`public static void main(String[])`

Das Programm soll die Eingabe auf `stdin` zeilenweise einlesen und die Zeilen dann nummeriert auf `stdout` ausgeben. Über Kommandozeilenargumente soll angegeben werden können, welcher Zeilenbereich ausgegeben werden soll.

- Ohne Argumente werden alle Zeilen ausgegeben.
- Mit einem Argument n werden die Zeilen ab Zeile n ausgegeben. Ist n größer als die Anzahl der Zeilen in der Eingabe, soll gar nichts ausgegeben werden.
- Mit zwei Argumenten n und m werden die Zeilen von Zeile n bis Zeile m ausgegeben (soweit vorhanden). Ist $m < n$, oder n größer als die Anzahl der Zeilen in der Eingabe, soll gar nichts ausgegeben werden.

Zeilennummern sind immer positive, ganze Zahlen (also ≥ 1), die in den Datentyp `long` passen. Wird ihr Programm mit unsinnigen Argumenten aufgerufen, geben Sie folgende Fehlermeldung aus (auf `stdout`)):

```
Bad arguments.
Usage: line-numbers [start-number [end-number]]
```

Beispieleingabe 01 (ohne Argumente):

```
diese datei
wird wieder
ausgegeben
```

Erwartete Ausgabe 01:

```
1 diese datei
2 wird wieder
3 ausgegeben
```

Beispieleingabe 02 (Argumente: `"2"`):

```
diese datei
wird wieder
ausgegeben
```

Erwartete Ausgabe 02:

```
2 wird wieder
3 ausgegeben
```

Beispieleingabe 03 (Argumente: `"a" "2"`):

```
eins
zwei
drei
```

Erwartete Ausgabe 03:

```
Bad arguments.
Usage: line-numbers [start-number [end-number]]
```

Beispieleingabe 04 (Argumente: `"2" "2"`):

```
diese datei
wird wieder
ausgegeben
```

Erwartete Ausgabe 04:

```
2 wird wieder
```

Hinweise:

- Die Kommandozeilenargumente sind im Parameter der `main` Methode als `String` Array gespeichert (typischerweise nennen wir den Parameter daher `args`).
- In IntelliJ lassen sich folgendermaßen Kommandozeilenargumente angeben:
  - Den „Run Configurations" öffnen mit `Run -> Edit Configurations ...`
  - Unter „Program arguments" die gewünschten Argumente eingeben. Mehrere Argumente werden durch Leerzeichen getrennt.

  Weitere Informationen: `https://docs.oracle.com/javase/tutorial/essential/environment/cmdLineArgs.html`
- In den Test-Dateien sind die Argumente im Dateinamen angegeben, durch Punkte („.") getrennt. Beispiele:
  - Der Testinput `line-numbers-02.4.stdin` übergibt das Argument `4`.
  - Der Testinput `line-numbers-03.2.a.stdin` übergitb die Argumente `2` und `a`.
- Wenn `arr` ein Array ist, dann gibt der Ausdruck `arr.length` die Länge von `arr` zurück. (Beachten Sie die fehlenden Klammern) Weiter Informationen zu Java-Arrays gibt es hier: `http://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html`
- Die nützliche `Scanner` Klasse lässt sich auch mit `String`s instanziieren: `https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html#Scanner(java.lang.String)`

Programmieren in Java

http://proglang.informatik.uni-freiburg.de/teaching/java/2017/

second-highest

Die zweithöchste Zahl

Woche 03 Aufgabe 2/4

Herausgabe: 2017-05-08
Abgabe: 2017-05-19

Achtung: beachten Sie unbedingt die allgemeinen Hinweise zur Abgabe auf der Homepage.

Project `second-highest`
Package `secondhighest`
Klassen

`Main`
`public static int secondHighest(int[] numbers)`

Implementieren Sie die Funktion `secondHighest`, die in einem `int`-Array die *zweit*größte Zahl findet und zurückgibt. Kann die Zahl nicht gefunden werden, soll `Integer.MIN_VALUE` zurückgegeben werden.

Ihre Implementierung sollte das Ergebnis in nur einem Durchlauf des Arrays `numbers` berechnen (also ohne das Array beispielsweise zu sortieren). Diese Anforderung zählt zur Code-Qualität!

Beispielaufruf 01: `Main.secondHighest(new int[]{1, 3, 3, 2})` ergibt `2` (als `int`)

Beispielaufruf 02: `Main.secondHighest(new int[]{})` ergibt `-2147483648` (als `int`)

Programmieren in Java

`http://proglang.informatik.uni-freiburg.de/teaching/java/2017/`

maximum-hourglass

Die größte Sanduhr

Woche 03 Aufgabe 3/4

Herausgabe: 2017-05-08
Abgabe: 2017-05-19

Achtung: beachten Sie unbedingt die allgemeinen Hinweise zur Abgabe auf der Homepage.

Project `maximum-hourglass`
Package `maximumhourglass`
Klassen

`Main`
`public static int maxHourglass(int[][] matrix);`

Implementieren Sie eine Methode, die aus einem 6x6 2D Array von `int`s die *Sanduhr* mit der größten Summe findet. Betrachten Sie zum Beispiel folgendes Array M:

$$M = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Die drei obersten linken Sanduhren in M sind:

$$S1 = \begin{bmatrix} 1 & 1 & 1 \\ & 1 & \\ 1 & 1 & 1 \end{bmatrix}$$

$$S2 = \begin{bmatrix} 1 & 1 & 0 \\ & 0 & \\ 1 & 1 & 0 \end{bmatrix}$$

$$S3 = \begin{bmatrix} 1 & 0 & 0 \\ & 1 & \\ 1 & 0 & 0 \end{bmatrix}$$

Die Sanduhren haben jeweils die Summen 7, 4 und 2. In diesem Fall sollte `maxHourglass` also 7 als die höchste Summe auf `stdout` zurückgeben.

Sie können davon ausgehen, dass nur 6x6 Arrays an Ihre Methode übergeben werden.

Programmieren in Java

`http://proglang.informatik.uni-freiburg.de/teaching/java/2017/`

caesar-cipher

Caesar-Verschlüsselung

Woche 03 Aufgabe 4/4

Herausgabe: 2017-05-08
Abgabe: 2017-05-19

Achtung: beachten Sie unbedingt die allgemeinen Hinweise zur Abgabe auf der Homepage.

Project `caesar-cipher`
Package `caesarcipher`
Klassen

`Main`
`public static String encode(String input, int shift)`

Die Aufgabe besteht darin eine Caesar-Verschlüsselung zu implementieren.

`https://en.wikipedia.org/wiki/Caesar_cipher`

Beachten Sie dabei:

- `encode` soll nur die 26 Buchstaben des Alphabets verschlüsseln (`a-z`, bzw. `A-Z`). Enthält `input` weitere Zeichen, bleiben diese unverändert.
- Die Groß- und Kleinschreibung soll beim Verschlüsseln erhalten bleiben.
- `encode` wird nur mit `shift`≥ 0 aufgerufen.

Beispielaufruf 01: `Main.encode("middle-Outz", 2)` ergibt `okffng-Qwvb` (als `String`)

Hinweise

- Die Klasse `java.lang.StringBuilder` eignet sich gut um einen String Zeichen für Zeichen aufzubauen.
- Der folgende Code demonstriert wie man mit sogenannten *Typecasts* in Java zwischen `char`s (Zeichen) und `int`s konvertieren kann:

```
char a = 'a';
int intOfChar = (int)a;
char charOfInt = (char)intOfChar;
// es gilt: a == charOfInt
```

  Die Typecast-Operatoren sind hier `(int)` bzw `(char)`. Ein Typecast konvertiert einen Wert in den in den in Klammern angegebenen Typ.
- Die Kodierung der Buchstaben des Alphabets in Java (UTF-8) entspricht der ASCII Kodierung.