W02P02 - Kontrollstrukturen I

Präsenzaufgabe

Submission due: 5 months ago

Points: 6 of 6 Optional Assessment: automatic

Practice Clone Repository

100% (6 months ago) GRADED

Recent results:

66.67% (6 months ago) GRADED
83.33% (6 months ago) GRADED
100% (6 months ago) GRADED
100% (6 months ago) GRADED
100% (6 months ago) GRADED

Show all results

Tasks:

Kontrollstrukturen I

Implementiere folgende sechs Funktionen in der Datei `ControlStructuresI` im Code-Template dieser Aufgabe. Verwende dabei keine Java-Bibliotheken wie `Math`. Die Befehle `System.out.print()` *(übergebenen Text auf Konsole ausgeben)* und `System.out.println()` *(übergebenen Text auf Konsole ausgeben und Zeilenumbruch - "\n" - am Ende anfügen)* bilden hierfür Ausnahmen. Sie dürfen (und müssen) verwendet werden. Verwende außerdem keine `for`-Schleifen. Die sind dann in der nächsten Aufgabe dran. Nutze die vorgegebene `main()`-Methode zum Testen deines Codes. Sie muss für vollständiges Lösen dieser Aufgabe keinen speziellen Inhalt haben.

Collatz-Folge 1 of 1 tests passing

Die Collatz-Folge ist eine mathematische Folge, die wie folgt definiert ist:

- Beginne mit irgendeiner natürlichen Zahl $n > 0$.
- Ist n gerade, so nimm als nächstes $\frac{n}{2}$.
- Ist n ungerade, so nimm als nächstes $3 * n + 1$.
- Wiederhole die Vorgehensweise mit der erhaltenen Zahl.

Die bislang unbewiesene Collatz-Vermutung besagt, dass diese Folge für alle natürlichen Zahlen n im Zyklus $(4, 2, 1)$ mündet.

Ergänze die Methode `printCollatz()` in `ControlStucturesI.java` so, dass diese die von dem Parameter n ausgehende Collatz-Folge berechnet, bis die 1 erreicht wurde. Dein Programm soll alle Zahlen der Folge durch Leerzeichen getrennt am Bildschirm ausgeben. Ist die eingegebene Zahl $n \leq 0$, soll "`Eingabe_muss_größer_als_0_sein!`" ausgegeben werden. Zusätzlich soll dein Programm die Länge der Folge mitzählen und abschließend in einer neuen Zeile "`Länge:_`" sowie die Länge der Folge ausgeben.

Hinweis: Benutze `System.out.print()`, um einen Text auf der Konsole ohne Zeilenumbruch danach und `System.out.println()` um einen Text auf der Konsole mit Zeilenumbruch danach auszugeben. Probiere die beiden Befehle in der `main()`-Methode aus!

Beispiele

1. Ausgabe für Parameter 0:

```
Eingabe muss größer als 0 sein!
```

2. Ausgabe für Parameter 4:

```
4 2 1
Länge: 3
```

3. Ausgabe für Parameter 11:

```
11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
Länge: 15
```

4. Ausgabe für Parameter 27:

```
27 82 41 124 62 31 94 47 142 71 214 107 322 161 484 242 121 364 182 91 274 137 412 206 103 310 155 466 233 700 350 175 526 263
790 395 1186 593 1780 890 445 1336 668 334 167 502 251 754 377 1132 566 283 850 425 1276 638 319 958 479 1438 719 2158 1079
3238 1619 4858 2429 7288 3644 1822 911 2734 1367 4102 2051 6154 3077 9232 4616 2308 1154 577 1732 866 433 1300 650 325 976 488
244 122 61 184 92 46 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1
Länge: 112
```

Hinweis: Dein Programm soll alle Folgenglieder in einer Zeile ausgeben, die Zeilenumbrüche sind nur der Darstellung halber hier

Zweier-Potenzen 1 of 1 tests passing

Ergänze die Methode `printPowersOfTwoUpTo()` so, dass sie alle Zweierpotenzen (von 2^0 an), die kleiner gleich dem Parameter `n` sind, in aufsteigender Reihenfolge mit je einem Leerzeichen getrennt ausgibt. Am Ende der Textausgabe sollte kein Leerzeichen mehr stehen. Ist die eingegebene Zahl $n \leq 0$, soll "`Eingabe_muss_größer_als_0_sein!`" ausgegeben werden.

Beispiele:

1. Ausgabe mit Parameter 0:

```
Eingabe muss größer 0 sein!
```

2. Ausgabe mit Parameter 7:

```
1 2 4
```

3. Ausgabe mit Parameter 8:

```
1 2 4 8
```

4. Ausgabe mit Parameter 9:

```
1 2 4 8
```

5. Ausgabe mit Parameter 1 000 000:

```
1 2 4 8 16 32 64 128 256 512 1024 2048 4096 8192 16384 32768 65536 131072 262144 524288
```

Dreiecke aus Sternchen 1 of 1 tests passing

Ergänze die Methode `printTriangle()` so, dass sie auf der Konsole ein Dreieck aus '*'-Charakteren mit Seitenlänge `sideLength` ausgibt, wie in den Beispielen gezeigt.

Beispiele:

1. Ausgabe mit Parameter 0:

```
Eingabe muss größer als 0 sein!
```

2. Ausgabe mit Parameter 1:

```
*
```

3. Ausgabe mit Parameter 3:

```
***
**
*
```

4. Ausgabe mit Parameter 6:

```
******
*****
****
***
**
*
```

Anzahl Ziffern 1 of 1 tests passing

Ergänze die Methode `calculateNumberOfDigits()` so, dass sie für eine übergebene Zahl $n \geq 0$ die Anzahl an Ziffern (in dezimaler Schreibweise) dieser zurückgibt. Wie übergebene Werte < 0 von der Methode behandelt werden, ist irrelevant. Die Tests überprüfen solche Werte nicht.

Beispiele:

1. Für Eingabe `0` sollte die Zahl `0` zurückgegeben werden.
2. Für Eingabe `7` sollte die Zahl `1` zurückgegeben werden.
3. Für Eingabe `37` sollte die Zahl `2` zurückgegeben werden.
4. Für Eingabe `1 234 567` sollte die Zahl `7` zurückgegeben werden.

Zahlen Umdrehen 1 of 1 tests passing

Ergänze die Methode `reverseNumber()` so, dass diese eine übergebene Zahl $0 \leq n \leq 999\ 999\ 999$ umdreht. D.h. es soll die Zahl zurückgegeben werden, die im Dezimalsystem mit genau der gleichen Ziffernfolge, nur in umgekehrter Reihenfolge dargestellt wird. Falls die übergebene Zahl in Nullen endet, sollen diese als führende Ziffern der umgekehrten Zahl weggelassen werden. D.h. 10 kehrt sich zu 01 um, was 1 entspricht, wenn man die führenden Nullen weglässt. Ebenso kehrt sich 100 zu 1 um. Und 1000 und 10000 usw. Wie übergebene Werte außerhalb des gegebenen Bereiches von der Methode behandelt werden, ist irrelevant. Die Tests überprüfen solche Werte nicht.

Beispiele:

1. Für Eingabe `0` soll der Rückgabewert `0` produziert werden.
2. Für Eingabe `5` soll der Rückgabewert `5` produziert werden.
3. Für Eingabe `127` soll der Rückgabewert `721` produziert werden.
4. Für Eingabe `6 148 229` soll der Rückgabewert `9 228 416` produziert werden.
5. Für Eingabe `1 200` soll der Rückgabewert `21` produziert werden.

Palindrome 1 of 1 tests passing

Ergänze die Methode `isPalindrome()` so, dass diese `true` zurückgibt, wenn die übergebene Zahl in Dezimaldarstellung ein Palindrom ist und `false`, wenn nicht. Ein Palindrom ist eine Zeichenfolge (hier: Ziffernfolge), die von vorwärts und rückwäarts gelesen gleich sind.

Beispiele:

1. Für folgende Eingaben sollte `true` zurückgegeben werden: `0`, `1`, `7`, `22`, `616`, `5005`, `1 234 554 321`
2. Für folgende Eingaben sollte `false` zurückgegeben werden: `21`, `264`, `5015`, `1 212 121 212`

FAQ

Q: Wofür ist das FAQ?

A: Um häufig gestellte Fragen zu sammeln. Wer trotzdem noch fragt macht Pinguine traurig.

Q: Was bedeutet `org.opentest4j.AssertionFailedError:[...] ==> expected: <2> but was: <1>`

A: Eine Zeile gilt nur als beendet, wenn ein sogenanntes Endzeichen kommt. `\n` Kann in einem String hinzugefügt werden um einen Zeilenumbruch zu verurschen. Die oben genannte Fehlermeldung bedeutet dass ihr eine Zeile zu wenig ausgebt. Dies liegt vermutlich daran, dasss ihr `System.out.print()` verwendet welches im Vergleich zu `println()` keinen Zeilenumbruch ausgibt.

Lösungsvorschlag