W02P03 - Kontrollstrukturen II

Präsenzaufgabe

Submission due: 5 months ago

Points: 4 Optional Assessment: automatic

Practice

You have missed the deadline.

Tasks:

Kontrollstrukturen II

Implementiere die folgenden vier Funktionen in der Datei `ControlStructuresII.java`!

Three and Seven No results

Vervollständige die Methode `threeAndSeven()` so, dass diese die Summe aller positiven Zahlen, die durch 3 oder 7 teilbar und kleiner oder gleich einer gegebenen natürlichen Zahl `n` sind, berechnet und zurückgibt. Wird eine Zahl < 0 übergeben, soll dies durch eine Konsolen-Ausgabe bemerkt werden: "`Eingabe_muss_größer_oder_gleich_0_sein!`"

Wird z.B. die Zahl 25 übergeben, müssen die Zahlen 3, 6, 7, 9, 12, 14, 15, 18, 21 und 24 summiert werden, was 129 ergibt.

Beispiele:

1. Eingabe `-2`: Produziert die Konsolen-Ausgabe

```
Eingabe muss größer oder gleich 0 sein!
```

2. Eingabe `7` ⟶ Ausgabe `16`
3. Eingabe `25` ⟶ Ausgabe `129`
4. Eingabe `92` ⟶ Ausgabe `1822`

ASCII No results

Vervollständige die Methode `printAsciiCodesFor()`. Diese soll für den Character `start` den ASCII-Code mit dem Satz "`Der_ASCII-Code_von_'`>dann der Character>"`'_ist_.`" in einer Zeile ausgeben. Dies soll es auch für den nächsten Character (mit dem nächstgrößeren ASCII-Code) machen, bis für `count` Character der ASCII-Code ausgegeben wurde.

Beispiele:

1. `printAsciiCodesFor('a', 5)` soll folgende Ausgabe produzieren:

```
Der ASCII-Code von 'a' ist 97.
Der ASCII-Code von 'b' ist 98.
Der ASCII-Code von 'c' ist 99.
Der ASCII-Code von 'd' ist 100.
Der ASCII-Code von 'e' ist 101.
```

2. `printAsciiCodesFor('X', 10)` soll folgende Ausgabe produzieren:

```
Der ASCII-Code von 'X' ist 88.
Der ASCII-Code von 'Y' ist 89.
Der ASCII-Code von 'Z' ist 90.
Der ASCII-Code von '[' ist 91.
Der ASCII-Code von '\' ist 92.
Der ASCII-Code von ']' ist 93.
Der ASCII-Code von '^' ist 94.
Der ASCII-Code von '_' ist 95.
Der ASCII-Code von '`' ist 96.
Der ASCII-Code von 'a' ist 97.
```

3. `printAsciiCodesFor('*', 8)` soll folgende Ausgabe produzieren:

```
Der ASCII-Code von '*' ist 42.
Der ASCII-Code von '+' ist 43.
```

```
Der ASCII-Code von ',' ist 44.
Der ASCII-Code von '-' ist 45.
Der ASCII-Code von '.' ist 46.
Der ASCII-Code von '/' ist 47.
Der ASCII-Code von '0' ist 48.
Der ASCII-Code von '1' ist 49.
```

Folgende ASCII-Tabelle mag für's Testen der Implementierung nützlich sein.

? Multiplikations-Tabelle No results

Vervollständige die Methode `printMultiplicationTable()` so, dass diese bei Eingabe `n` eine n*n-Multiplikations-Tabelle wie in den Beispielen gezeigt ausgibt. Die Tabellen müssen nur bis `n = 30` korrekt formatiert sein. Es reicht also, wenn du nach jeder Zahl/jedem Zeichen in der Tabelle zum nächsten Tab springst. Verwende hierzu den Character `\t`.

Da die genaue Anzahl an Leerzeichen in der Tabelle umständlich zu beschreiben bzw. die Tabelle bei ungenauer Beschreibung schwierig zu testen wäre, gibt es für diese Teilaufgabe keine Unit-Tests. Du solltest allerdings nach Augenmaß leicht feststellen können, ob deine Tabelle hübsch formatiert ist oder nicht. Eine übersichtliche Formatierung zu erhalten mag ein bisschen Fingerspitzengefühl und Fine-Tuning der genauen Abstände/Anzahl an Strichen verlangen. Wie ihr anhand des zweiten Beispiels seht, kann das selbst Artemis nicht so richtig ;-)

Beispiele:

1. `printMultiplicationTable(2)` sollte folgende Ausgabe produzieren:

```
*    |   1   2
---------------
1    |   1   2
2    |   2   4
```

2. `printMultiplicationTable(15)` sollte folgende Ausgabe produzieren:

```
*    |   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
---------------------------------------------------------------------
1    |   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15
2    |   2   4   6   8   10  12  14  16  18  20  22  24  26  28  30
3    |   3   6   9   12  15  18  21  24  27  30  33  36  39  42  45
4    |   4   8   12  16  20  24  28  32  36  40  44  48  52  56  60
5    |   5   10  15  20  25  30  35  40  45  50  55  60  65  70  75
6    |   6   12  18  24  30  36  42  48  54  60  66  72  78  84  90
7    |   7   14  21  28  35  42  49  56  63  70  77  84  91  98  105
8    |   8   16  24  32  40  48  56  64  72  80  88  96  104 112 120
9    |   9   18  27  36  45  54  63  72  81  90  99  108 117 126 135
10   |   10  20  30  40  50  60  70  80  90  100 110 120 130 140 150
11   |   11  22  33  44  55  66  77  88  99  110 121 132 143 154 165
12   |   12  24  36  48  60  72  84  96  108 120 132 144 156 168 180
13   |   13  26  39  52  65  78  91  104 117 130 143 156 169 182 195
14   |   14  28  42  56  70  84  98  112 126 140 154 168 182 196 210
15   |   15  30  45  60  75  90  105 120 135 150 165 180 195 210 225
```

? Primzahlen Ausgeben No results

Vervollständige die Methode `printPrimesUpTo()` so, dass für eine übergebene Zahl `n` alle Primzahlen kleiner gleich dieser Zahl auf der Konsole - mit je einem Leerzeichen getrennt - ausgegeben werden.

Beispiele:

1. `printPrimesUpTo(1)` sollte folgende Konsolenausgabe produzieren:

```

```

2. `printPrimesUpTo(7)` sollte folgende Konsolenausgabe produzieren:

```
2 3 5 7
```

3. `printPrimesUpTo(100)` sollte folgende Konsolenausgabe produzieren:

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

Lösungsvorschlag

Tests