SPENGERGASSE

3BKIF, 2. Praktische Leistungsfeststellung

14. Dezember 2023, Lehrer: Georg Graf

Vorbemerkung zu Testklassen

Da Sie bei dieser Leistungsfeststellung alle Hilfsmittel verwenden dürfen, ist in den Testklassen jeweils der tatsächliche Test enfernt worden, jedoch nicht die Rümpfe der Methoden, somit können Sie erkennen, was bei der Auswertung der PLF überprüft werden wird. Es funktionieren also gleich in der Angabe alle Tests. Lassen Sie sich davon nicht täuschen!

Sie brauchen keine Tests zu erstellen!

1. Aufgabe: Methode *drawGrid(lines, columns)*

Die Methode möge ein Raster zeichnen, mit sovielen `lines` und `columns` wie es in den Parametern übergeben wurde. In den Raster werden horizontale und vertikale Linien eingezeichnet. Bei den horizontalen Linien soll nur jede zweite gezeichnet werden, bei den vertikalen nur jede dritte. Auch Randphänomene wie nur eine Zeile oder auch eine Spalte sollen richtig verarbeitet werden.

Achtung: Der erzeugte String **darf am Zeilenende keine Leerzeichen haben**, sonst schlagen die Tests fehl. Beachten Sie also völlig exakt, ob ein Leerzeichen, ein Punkt oder eine Raute ausgegeben nicht. Die Testklasse läßt hier keine Abweichungen zu!! Bedienen Sie sich der Java Online Dokumentation zur String-Klasse und bedenken Sie das Wort "trailing".

Die Methode soll den erzeugten String

1. auf die Konsole ausgeben
2. an den Aufrufer retournieren

Beispiel (lines: 7, columns: 7)

```
# # # # # # #
# . . # . . #
# # # # # # #
# . . # . . #
# # # # # # #
# . . # . . #
# # # # # # #
```

Beispiel (lines: 5, columns: 10)

```
# # # # # # # # # #
# . . # . . # . . #
# # # # # # # # # #
# . . # . . # . . #
# # # # # # # # # #
```

Beispiel (lines: 5, columns: 7

```
# # # # # # #
# . . # . . #
# # # # # # #
# . . # . . #
# # # # # # #
```

2. Aufgabe *Parameterprüfung*

Wie aus den obigen Beispielen ersichtlich, machen nicht alle Eingaben Sinn. Bei `lines == 4` würde das Raster z.B. ohne Bodenlinie enden, bei `columns == 8` z.B. würde der Grid an der rechten Kante unschön aussehen. Implementieren Sie 2 Parameterprüfungen (für zeile und spalte), welche gewährleisten, daß nur "schöne" Formen ausgegeben werden. Im Falle einer nicht bestandenen Parameterprüfung werfen Sie bitte eine `IllegalArgumentException` mit einer entspechenden Message.

3. Aufgabe *befülltes Array zurückgeben*

Gewünscht ist in dieser Aufgabe eine Funktion, welche ein `long []` Array

1. erstellt
2. mit Zahlen befüllt (s.u.)
3. retourniert

Für die Befüllung des Arrays werden sog. Fibonacci-Zahlen gewünscht, wie folgt:

Die erste Fibonacci-Zahl lautet `0`, die zweite `1`. Jede weitere Fibonacci-Zahl wird durch Addition ihrer beiden Vorgänger ermittelt, somit wäre die Rückgabe von z.B. `getFirstFibonaccis(6)` das Array `[0, 1, 1, 2, 3, 5]`.

4. Aufgabe Parameterprüfung

Stellen Sie sicher, daß der Aufruf von `getFirstFibonaccis(int n)` nur für n im Bereich von `0 ... 30` eine Antwort retourniert. Im anderen Fall soll wieder eine `IllegalArgumentException` aufgeworfen werden.

Gutes Gelingen!