Softwareentwicklung I (IB)

Blatt 4

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 28.01.2020 17:18

Ab diesem Blatt führen unsinnige oder nichts sagende Commit-Messages zum Nichtbestehen der jeweiligen Aufgabe!

Abgabe aller Aufgaben auf diesem Blatt: bis 13.12.17 10:00 Uhr durch Pushen in das zur Aufgabe gehörende GitHub-Repository.

Aufgabe 1

Das Repository für diese Aufgabe bekommen Sie unter https://classroom.github.com/a/sSsyz0VZ.

Schreiben Sie ein Programm `Chessboard` das eine ganze Zahl als Kommandozeilenparameter akzeptiert und ein Schachbrett dieser Größe ausgibt.

Statt der Farben Weiß und Schwarz verwenden Sie die Symbole `-` und `+`. Das erste ausgegebene Symbol muss ein `-` sein.

Die Symbole wechseln dann jeweils ab, so dass nie zwei gleiche Symbole direkt neben- oder untereinander stehen. Geben Sie nach jedem Symbol **außer** dem letzten in einer Zeile ein Leerzeichen aus.

Im Folgenden sehen Sie die Aufrufe und Ausgaben für die Kommandozeilenparameter 0 bis 8:

```
$ ./gradlew run -Dexec.args="0"
$ ./gradlew run -Dexec.args="1"
-
```

```
$ ./gradlew run -Dexec.args="2"
- +
+ -
$ ./gradlew run -Dexec.args="3"
- + -
+ - +
- + -
$ ./gradlew run -Dexec.args="4"
- + - +
+ - + -
- + - +
+ - + -
$ ./gradlew run -Dexec.args="5"
- + - + -
+ - + - +
- + - + -
+ - + - +
- + - + -
$ ./gradlew run -Dexec.args="6"
- + - + - +
+ - + - + -
- + - + - +
+ - + - + -
- + - + - +
+ - + - + -
$ ./gradlew run -Dexec.args="7"
- + - + - + -
+ - + - + - +
- + - + - + -
+ - + - + - +
- + - + - + -
+ - + - + - +
- + - + - + -
$ ./gradlew run -Dexec.args="8"
- + - + - + - +
+ - + - + - + -
- + - + - + - +
+ - + - + - + -
- + - + - + - +
+ - + - + - + -
- + - + - + - +
+ - + - + - + -
```

Randbedingungen:

- Verwenden Sie zur Realisierung `while`-Schleifen

Die Tests, deren Name mit `chessboard_` beginnen, sollten jetzt durchlaufen.

Um die Tests, die mit dem Namen `chessboardWithX_` beginnen, auch zu erfüllen, müssen Sie Ihre Lösung wie folgt erweitern:

Erweitern Sie Ihr Programm `Chessboard` so, dass Sie **optional** 2 weitere Kommandozeilenparameter übergeben können. Der erste ist eine Zeilennummer, der zweite eine Spaltenummer. Genau an dieser Koordinate soll statt einem `-` oder `+` ein `X` stehen. Liegt die Koordinate außerhalb des Schachbrettes wird sie ignoriert.

Hinweis: Mit `args.length` können Sie die Länge des `args`-Arrays herausbekommen. Das Programm soll nach wie vor auch mit nur einem Kommandozeilenparameter funktionieren.

Beispiele:

```
$ ./gradlew run -Dexec.args="8 3 7"
- + - + - + - +
+ - + - + - + -
- + - + - + X +
+ - + - + - + -
- + - + - + - +
+ - + - + - + -
- + - + - + - +
+ - + - + - + -
$ ./gradlew run -Dexec.args="8 3 70"
- + - + - + - +
+ - + - + - + -
- + - + - + - +
+ - + - + - + -
- + - + - + - +
+ - + - + - + -
- + - + - + - +
+ - + - + - + -
$ ./gradlew run -Dexec.args="5 3 3"
- + - + -
+ - + - +
- + X + -
+ - + - +
- + - + -
```

Aufgabe 2

Das Repository für diese Aufgabe bekommen Sie unter https://classroom.github.com/a/tksQGFma.

Schreiben Sie ein Programm `ChristmasTree` das eine ganze Zahl als Kommandozeilenparameter akzeptiert und einen Weihnachtsbaum in der Gesamthöhe der eingegebenen Zahl ausgibt.

Der Weihnachtsbaum besteht aus dem Zeichen `*` für die Äste respektive Tannennadeln und dem Zeichen `I` für den Stamm. Der Stamm hat immer die Höhe 1 und zählt mit zur Gesamthöhe.

Die Spitze des Baumes besteht aus einem `*`, die zweite Zeile aus drei, die dritte aus fünf und so weiter. Der Baum ist symmetrisch und stösst mit der ersten Zeile über dem Stamm an den linken Rand an, d.h. diese Zeile beginnt mit einem `*` und nicht mit einem Leerzeichen.

Der Stamm ist bis zu einer Höhe von 9 genau ein Zeichen breit. Zwischen 10 und 19 ist er drei Zeichen breit, zwischen 20 und 29 ist er fünf Zeichen breit, usw.

Der Baum soll erst ab einer Mindesthöhe von 3 ausgegeben werden. Andernfalls geben Sie `zu klein` aus.

Beispielaufrufe sind

```
$ ./gradlew run -Dexec.args="2"
zu klein
$ ./gradlew run -Dexec.args="-5"
zu klein
$ ./gradlew run -Dexec.args="3"
 *
***
 I
$ ./gradlew run -Dexec.args="6"
    *
   ***
  *****
 *******
*********
    I
$ ./gradlew run -Dexec.args="10"
        *
       ***
      *****
     *******
    *********
```

```
   ***********
  ************
 **************
****************
       III
$ ./gradlew run -Dexec.args="25"
                        *
                       ***
                      *****
                     *******
                    *********
                   ***********
                  ************
                 **************
                ****************
               ******************
              ********************
             **********************
            ************************
           **************************
          ****************************
         ******************************
        ********************************
       **********************************
      ************************************
     **************************************
    ****************************************
   ******************************************
  ********************************************
 **********************************************
**************************************************
                     IIIII
```

Randbedingungen:

- Verwenden Sie zur Realisierung `for`-Schleifen

Aufgabe 3

Das Repository für diese Aufgabe bekommen Sie unter https://classroom.github.com/a/L3ZnnjhR.

Leonardo Fibonacci hat 1202 das Wachstum einer Kaninchenpopulation nach folgenden Regeln beschrieben:

1. Jedes Paar Kaninchen wirft pro Monat ein weiteres Paar Kaninchen.
2. Ein neugeborenes Paar bekommt erst im zweiten Lebensmonat Nachwuchs (die Austragungszeit reicht von einem Monat in den nächsten).
3. Die Tiere befinden sich in einem abgeschlossenen Raum, so dass kein Tier die Population verlassen und keines von außen hinzukommen kann.

Fibonacci begann die Reihe, nicht ganz konsequent, nicht mit einem neugeborenen, sondern mit einem trächtigen Paar, das seinen Nachwuchs bereits im ersten Monat wirft, so dass im ersten Monat bereits 2 Paare zu zählen sind. In jedem Folgemonat kommt dann zu der Anzahl der Paare, die im Vormonat gelebt haben, eine Anzahl von neugeborenen Paaren hinzu, die gleich der Anzahl derjenigen Paare ist, die bereits im vorvergangenen Monat gelebt hatten, da der Nachwuchs des Vormonats noch zu jung ist, um jetzt schon seinerseits Nachwuchs zu werfen. Fibonacci führte den Sachverhalt für die zwölf Monate eines Jahres vor (2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377) und wies auf das Bildungsgesetz der Reihe durch Summierung jeweils zweier aufeinanderfolgender Reihenglieder (2+3=5, 3+5=8, 5+8=13 usw.) hin. Er merkte außerdem an, dass die Reihe sich nach diesem Prinzip für eine unendliche Zahl von Monaten fortsetzen lässt, was dann allerdings unsterbliche Kaninchen voraussetzt.

Die nach Leonardo Fibonacci benannte Fibonacci-Folge ist für natürliche Zahlen wie folgt definiert:

```
fib(1) = 1
fib(2) = 1
fib(n) = fib(n-1) + fib(n-2)
```

Schreiben Sie ein Programm `Fibonacci` das eine ganze Zahl als Kommandozeilenparameter akzeptiert und die jeweils zugehörige Fibonacci-Zahl ausgibt. Für einen negativen Parameter geben Sie als Ergebnis `-1` aus. Für die `0` soll `0` ausgegeben werden.

Randbedingungen:

- Sie dürfen im gesamten Programm **genau eine** `System.out.printf`-Anweisung nutzen.
- `System.out.print` und `System.out.println` sind verboten.
- In der `printf`-Anweisung dürfen Sie die Variablen **nicht** in den Format-String schreiben.
- Gestalten Sie die Ausgabe **genau** wie in den folgenden Beispielaufrufen angegeben.

Beispielaufrufe:

```
$ ./gradlew run -Dexec.args="-2"
-1
$ ./gradlew run -Dexec.args="0"
0
$ ./gradlew run -Dexec.args="1"
1
```

```
$ ./gradlew run -Dexec.args="2"
1
$ ./gradlew run -Dexec.args="3"
2
$ ./gradlew run -Dexec.args="4"
3
$ ./gradlew run -Dexec.args="5"
5
$ ./gradlew run -Dexec.args="6"
8
```