Softwareentwicklung I (IB)

Blatt 5

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik
Hochschule München

Letzte Änderung: 30.11.2017 21:38

Abgabe der Aufgabe auf diesem Blatt: bis 10.01.18, 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/8AVoQiWb.

In dieser Aufgabe sollen Sie den `ChristmasTree` vom letzten Blatt auf drei verschiedene Arten erweitern. Zum einen soll es möglich sein Kerzen (oder Kugeln) darauf gleichmäßig zu verteilen, in einer anderen Erweiterung sollen Sie den Baum einrahmen und die dritte Erweiterung soll den Baum vervielfältigen.

Gesteuert wird die die Ausgabe durch Kommandozeilenparameter. Dabei gibt es folgende Fälle:

1. Erster Kommandozeilenparameter `help`, Rest wird ignoriert:

   Es wird folgende Info zur Nutzung des Programmes ausgegeben und das Programm sofort beendet, also kein Baum ausgegeben.

   ```
   Usage: java ChristmasTree <height> <option>
   Options: candles <number of candles> <candle character>
            frame <frame character>
            clone <number of trees (original + clones)>
   ```

2. Erster Kommandozeilenparameter ist eine Zahl ≤ 2:

   Es wird `too small` ausgegeben und das Programm sofort beendet.

3. Erster Kommandozeilenparameter ist eine Zahl ≥ 3 und der zweite Kommandozeilenparameter ist etwas anderes als `candles`, `frame` oder `clone`:

   Es wird ausgegeben `Unknown option:` und das eingegebene Argument. Das Programm danach sofort beendet.

4. Erster Kommandozeilenparameter ist eine Zahl ≥ 3, zweiter Parameter ist `candles`, `frame` oder `clone`:

   Siehe Spezifikation von Kerzen, Rahmen bzw. Clone. Es ist immer nur eine Option möglich, also **entweder** Kerzen oder Rahmen oder Clones.

Kerzen

Mit der Option `candles` sollen Kerzen o.ä. gleichmäßig auf dem Baum verteilt werden. Für die Option `candles` werden 4 Argumente auf der Kommandozeile benötigt:

1. Die Höhe des Baumes, inkl. Stamm, wie auf Blatt 4, als `int`.
2. Die exakte Zeichenkette `candles`.
3. Die Anzahl der Kerzen die auf dem Baum verteilt werden sollen, als `int`.
4. Ein einzelnes Zeichen (`char`) das die Kerzen o.ä. repräsentieren soll, z.B. `i` oder `0`.

Anmerkung: Ein einzelnes Zeichen können Sie aus einer Zeichenkette mit der Methode `charAt` extrahieren. Argument ist dabei der bei 0 beginnende Index des Zeichens.

Verteilen Sie die Kerzen nach dem folgenden Schema von der Christbaumspitze aus:

- Vor der ersten Kerze und zwischen je zwei benachbarten Kerzen ist immer die gleiche Anzahl `x` von Sternen. Nach der letzten Kerze sind es höchstens `x`.
- Nicht in allen Fällen muss die Anzahl der übergebenen Kerzen wirklich aufgehen. Verteilen Sie dann **höchstens** `y` Kerzen.

  Wenn Sie beispielsweise einen Baum der Höhe 5 erzeugen, so hat dieser `1 + 3 + 5 + 7 = 16` Stellen. Wenn Sie nun 6 Kerzen verteilen wollen, bleiben 10 Sterne. Wenn Sie die Anzahl Sterne vor und zwischen zwei benachbarten Kerzen auf 1 setzen, sind nach der letzen Kerze noch 4 Sterne übrig. Das sieht nicht schön aus. Wenn Sie stattdessen 2 Sterne dazwischen Platz lassen, können Sie nur 5 Kerzen verteilen. Das sieht besser aus und ist so gewollt. Die sechste Kerze können Sie einfach ignorieren.

Rahmen

Mit der Option `frame` soll der Baum eingerahmt werden. Für die Option `frame` werden 3 Argumente auf der Kommandozeile benötigt:

1. Die Höhe des Baumes, inkl. Stamm, wie auf Blatt 4, als `int`.
2. Die exakte Zeichenkette `frame`.
3. Ein einzelnes Zeichen (`char`) aus dem der Rahmen bestehen soll, z.B. `#`.

Rahmen Sie Ihren Baum dann wie folgt:

- Der Rahmen besteht aus dem übergebenen Zeichen und umgibt den gesamten Baum.
- Der Rahmen berührt oben und unten den Baum.
- Zwischen dem Rahmen und der breitesten Stelle des Baumes ist jeweils ein Leerzeichen.

Clone

Mit der Option `clone` soll der Baum mehrfach nebeneinander ausgegeben werden. Für die Option `clone` werden 3 Argumente auf der Kommandozeile benötigt:

1. Die Höhe des Baumes, inkl. Stamm, wie auf Blatt 4, als `int`.
2. Die exakte Zeichenkette `clone`.
3. Die Anzahl wie oft der Baum nebeneinander stehen soll als `int`.

Geben Sie Ihre Bäume dann wie folgt aus:

- Sollen `0` Bäume ausgegeben werden, wird **gar nichts** ausgegeben. Auch keine Leerzeilen.
- Die Bäume müssen direkt **nebeneinander** stehen und an der breitesten Stelle ein Leerzeichen Abstand haben.

Beispielaufrufe (mit `gradlew`):

```
$ ./gradlew run -Dexec.args="help"
Usage: java ChristmasTree <height> <option>
Options: candles <number of candles> <candle character>
         frame <frame character>
         clone <number of trees (original + clones)>
$ ./gradlew run -Dexec.args="1"
too small
$ ./gradlew run -Dexec.args="2"
too small
$ ./gradlew run -Dexec.args="3 garnix"
Unknown option: garnix
```

```
$ ./gradlew run -Dexec.args="10 candles 25 i"
        *
       **i
      ***i*
     **i***i
    ***i***i*
   **i***i***i
  ***i***i***i*
 **i***i***i***i
***i***i***i***i*
       III
$ ./gradlew run -Dexec.args="9 candles 20 O"
       *
      **O
     ***O*
    **O***O
   ***O***O*
  **O***O***O
 ***O***O***O*
**O***O***O***O
       I

$ ./gradlew run -Dexec.args="8 frame #"
##################
#       *        #
#      ***       #
#     *****      #
#    *******     #
#   *********    #
#  ***********   #
# ************** #
#       I        #
##################
$ ./gradlew run -Dexec.args="7 frame %"
%%%%%%%%%%%%%%%
%       *       %
%      ***      %
%     *****     %
%    *******    %
%   *********   %
% *************
%       I       %
%%%%%%%%%%%%%%%
```

```
$ ./gradlew run -Dexec.args="7 clone 5"
     *           *           *           *           *
    ***         ***         ***         ***         ***
   *****       *****       *****       *****       *****
  *******     *******     *******     *******     *******
 *********   *********   *********   *********   *********
********** ********** ********** ********** **********
     I           I           I           I           I
$ ./gradlew run -Dexec.args="12 clone 3"
          *                   *                   *
         ***                 ***                 ***
        *****               *****               *****
       *******             *******             *******
      *********           *********           *********
     ***********         ***********         ***********
    *************       *************       *************
   ***************     ***************     ***************
  *****************   *****************   *****************
 *******************  *******************  *******************
*********************************************************
         III                 III                 III
```

Rahmenbedingungen und Hinweise

- Verwenden Sie zum Erkennen der Zeichenkette `help equals` und nutzen Sie für die drei Optionen `switch`.
- Strukturieren Sie Ihren Code so, dass Dinge die mehrfach benötigt werden nur einmal gemacht werden. Beispielsweise ist die Berechnung der Breite des Stammes für alle drei Baum-Optionen notwendig.
- Verwenden Sie nur eine einzige Ausgabe (aus Performanzgründen sinnvoll).

  Dazu bauen Sie den `String` schrittweise mit einem `StringBuilder` zusammen. Die Verwendung von `+` zum Konkatenieren von Zeichenketten ist **verboten**!

  Beispiel:

  ```
  // Erzeugen eines StringBuilders
  StringBuilder builder = new StringBuilder();
  // Aneinanderhängen von Zeichenketten mit dem StringBuilder
  builder.append("Hallo");
  builder.append(" ");
  builder.append("Welt\n");
  // Oder als sog. Aufrufkette
  builder.append("Hallo")
  ```

```
        .append(" ")
        .append("Welt\n");
// Ausgabe der gesamten Zeichenkette
System.out.println(builder);
```

- Verwenden Sie in Ihrem Code Kommentare die deutlich machen was gerade gemacht wird und helfen den Code und die Struktur schnell zu erfassen.

Und nun nur noch als Geschenk verpacken

Gradle bietet eine einfache Möglichkeit die entwickelte Applikation als sogenannte *Distribution* zu verpacken. Führen Sie dazu im Gradle-Projekt aus:

```
$ ./gradlew distZip
```

Anschließend finden Sie im Verzeichnis `build/distribution` eine Zip-Datei mit dem Namen `ChristmasTree.zip`. Diese Datei enthält alles was gebraucht wird um Ihr Programm **auszuführen**. Das einzige was außerdem auf dem Rechner installiert sein muss, ist eine Java-Laufzeit-Umgebung, also entweder ein JDK oder auch nur ein JRE.

Wenn Sie die Datei z.B. an Bekannte oder Verwandte per E-Mail verschicken, können diese die Datei entpacken und finden ein Verzeichnis `ChristmasTree`. Darin befindet sich das Verzeichnis `lib`— steht für Library/Bibliothek —in dem Ihr Programm und einige Abhängigkeiten als Java Archive (kurz JARs) enthalten sind. Im Verzeichnis `bin`— steht für Binary —sind zwei ausführbare Dateien: die Datei `ChristmasTree.bat` ist für Windows, die Datei `ChristmasTree` für Unix (macOS **ist** ein Unix). Analog zu den Dateien `gradlew` und `gradlew.bat` können diese ausgeführt werden. Wenn Sie z.B. im Verzeichnis `ChristmasTree` sind, starten Sie unter Windows z.B. mit

```
$ bin\ChristmasTree 12 frame +
```

bzw. unter Unix

```
$ bin/ChristmasTree 12 frame +
```

Der Quellcode ist in dem ZIP nicht enthalten, d.h. Sie könnten es so auch ohne die Sourcen verkaufen.

Um zu überprüfen was alles im ZIP enthalten ist, können Sie es übrigens ganz einfach mit dem Kommando

```
$ ./gradlew installDist
```

in das Verzeichnis `build\install` (Windows) bzw. `build/install` (Unix) installieren und dort ausführen.

Sie können das übrigens mit allen Ihren bisher erstellten Gradle-Projekten machen.