Programmierung 1
Prof. Dr. Jörg Kreiker
Wintersemester 2018/19

Hochschule Fulda
University of Applied Sciences

17. Januar 2019

Hausaufgabe 10

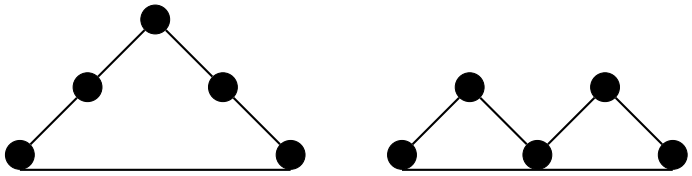
Abgabe: 28. Januar 2019, 23:59 Uhr

Abgabe

Füllt die vorgegebene Datei `Main.java` aus, wobei ihr die `main` Methode zum freien Testen verwenden dürft. Die Signaturen der Methoden bleiben unverändert. Gebt die Datei `Main.java` ab.

Aufgabe 1: Besoffenes Rentier (5 Punkte)

Rentier Rudolph hat einen über den Durst getrunken und muss nach Hause wanken. Er ist fast zu Hause und muss nur noch 22 Meter an einer Wand entlanggehen. Er braucht 22 Schritte, weil er mit jedem Schritt 1 Meter vorwärts kommt. Wegen des Glühweins schwankt er allerdings stets zusätzlich 1 Meter nach links (von der Wand weg) oder rechts (zur Wand hin). Er kann nur nach rechts schwanken, wenn er nicht schon an der Wand ist. Wie viele mögliche Wege kann Rudolph gehen, um nach 22 Metern am Ende der Wand zu sein? Beispiel: Bei einer 4 Meter langen Wand hätte er genau diese zwei Möglichkeiten.



Schreibt eine Methode

```
public static long rentier(int laenge, int x, int y)
```

die für eine Wand der Länge `laenge`, eine horizontale Position `x` und eine vertikale Position `y` die Anzahl der möglichen Torkelwege liefert. Die Startposition sei mit `x==0` und `y==0` kodiert. Die Endposition mit `x==laenge` und `y==0`. Der Aufruf `rentier(22,0,0)` liefert also das gesuchte Resultat. Die Basisfälle sind

- `y<0`: Hier gibt es 0 Möglichkeiten, weil Rudolph nicht durch die Wand kann
- Ebenfalls gibt es keine Möglichkeiten, wenn Rudolph *zu weit* von der Wand ist. Bei 22m kann er bei `x==11` höchstens auch `y==11` sein, wenn er danach nur noch nach rechts torkelt. Finden Sie das y-Limit für gegebenes `x` und `laenge`.
- Im Beispiel gibt es von $(21, 1)$ noch *eine* Möglichkeit und von $(21, y)$ für $y \neq 1$ *keine* Möglichkeit.

Aufgabe 2: Hanoi spezial (5 Punkte)

In dieser Aufgabe betrachten wir eine Variante der Türme von Hanoi, in der direkte Züge von Säule 1 zu Säule 3 verboten sind. Stattdessen muss man eine Scheibe in diesem Fall zuerst von 1 nach 2 und dann von 2 nach 3 bewegen. Man braucht also 2 Züge. Alle anderen Regeln bleiben gleich. Schreibt eine Methode

```
public static String hanoi3(int n, int von, int bis)
```

die die kürzeste Zugfolge ausgibt, um `n` Scheiben von `von` bis `bis` bewegt.