Prof. Dr. Matthias Hagen
Dipl.-Inform. Annett Thüring
Dipl.-Inform. Steffen Schiele

Martin-Luther-Universität Halle-Wittenberg
Institut für Informatik
Wintersemester 2019/ 2020

Übungsblatt 9: Objektorientierte Programmierung

Ausgabe: 09.01.2020 Abgabe: 16.01.2020

Aufgabe 1 (3 Punkte) Verwenden Sie den folgenden Freigabecode für die YAPEX-Aufgabe *Ackermann-Funktion*: 5d2b73g1lb95-86cb

Schreiben Sie ein Programm, welches als Kommandozeilenparameter zwei ganzzahlige Werte (im Folgenden als n und m bezeichnet) erhält und anschließend die ACKERMANN-Funktion berechnet und das Ergebnis ausgibt.

Die ACKERMANN-Funktion $\text{ack} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ ist wie folgt definiert:

$$\text{ack}(n, m) = \begin{cases} m + 1, & \text{falls } n = 0 \\ \text{ack}(n - 1, 1), & \text{falls } n > 0 \text{ und } m = 0 \\ \text{ack}(n - 1, \text{ack}(n, m - 1)), & \text{sonst.} \end{cases}$$

Aufgabe 2 (8 Punkte) Verwenden Sie den folgenden Freigabecode für die YAPEX-Aufgabe *Bruchklasse*: 5d2b73gk9c28-1a01

Erstellen Sie in der vorgegebenen Datei `Bruch.java` eine Klasse *Bruch*, mit der nicht negative rationale Zahlen dargestellt werden können und folgendes realisiert:

- Einen Konstruktor, der als Parameter den Zähler und Nenner als ganze Zahl erhält.
- Einen Konstruktor, welcher eine Zeichenkette erhält. Diese Zeichenkette repräsentiert einen Bruch, wobei garantiert ist, dass die übergeben Zeichenkette immer im Format `123/456` ist.
- Eine überladene *toString*-Methode, welche einen Bruch $\frac{3}{4}$ im Format `3/4` als String zurückgibt (ohne Leerzeichen und Zeilenumbruch!).
- Eine Instanzmethode *addiere*, welche als Parameter den zweiten Bruch erhält und als Rückgabewert den Ergebnisbruch zurückgibt (weder der aktuelle noch der übergebene Bruch werden verändert).
- Eine von außen nicht zugreifbare Methode *kuerze*, welche den aktuellen Bruch kürzt und zudem sicherstellt, dass der Bruch immer im gekürzten Zustand vorliegt.

Beachten Sie unbedingt, dass sämtliche Instanzvariablen nicht von außerhalb der Klasse veränderbar sein dürfen, Sie keine Bibliotheken benötigen sowie auch nicht importieren dürfen, und der Bruch sich zu jedem Zeitpunkt gekürzt ist. Zudem sollten die Konstruktoren, die *toString*- sowie *addiere*-Methode von jeder anderen Klasse aus zugreifbar sein, alle anderen Methoden dürfen nicht von außerhalb der Klasse zugreifbar sein..

Aufgabe 3 (10 Punkte) Verwenden Sie den folgenden Freigabecode für die YAPEX-Aufgabe *Labyrinth*: 5d2b73g64ipm-732c

Ziel dieser Aufgabe ist es ein Programm zu schreiben, welches ein Labyrinth von einer Datei liest und es einem Nutzer erlaubt sich in diesem Labyrinth zu bewegen. Dieser fängt immer in der ersten Zeile bei dem Durchgang an, der am weitesten links ist. Die Benutzereingaben werden bereits von einem Testprogramm eingelesen. Die Datei enthält in der ersten Zeile die Anzahl der Zeilen des Labyrinths und in der zweiten Zeile die Spalten des Labyrinths. Anschließend wird das Labyrinth bestehend aus den Zeichen # und `O` (Großbuchstabe O) dargestellt, wobei # für eine Wand und `O` für einen Durchgang steht. In der ersten Zeile befindet sich immer mindestens ein Durchgang.

Ihre Aufgabe ist es nun eine Klasse *Labyrinth*, welche intern eine Matrix vom Typ *boolean* verwaltet, zu schreiben. Die Matrix selbst soll nicht mehr verändert werden. Ihre Klasse darf darüber hinaus weitere Instanzvariablen besitzen, z. B. um sich zu merken wo der Spieler sich aktuell befindet. Beachten Sie, dass alle Instanzvariablen von außerhalb der Klasse nicht zugreifbar sein dürfen.

Implementieren Sie die folgenden Methoden:

- Einen Konstruktor, welcher als Parameter einen Dateinamen erhält, diese Datei einliest und entsprechend eine Matrix vom Typ *boolean* erstellt (*true* anstatt `O` und *false* anstatt #). Sie dürfen davon ausgehen, dass die Datei immer im oben angegebenen Format vorliegt. Anschließend muss der Konstruktor noch die aktuelle Position herausfinden. Diese befindet sich immer oben (in der ersten Zeile), bei dem ersten Durchgang (`O`) von links ausgesehen.
- Eine Methode *bewegeDich*, welche als Parameter ein Zeichen erhält und einen Wahrheitswert zurückgibt. Wenn die entsprechende Bewegung möglich ist, soll `true` und andernfalls `false` zurückgegeben werden. Eine Bewegung ist nicht möglich, wenn an der neuen Stelle ein Hindernis ist (Eintrag `false`) oder die Grenzen des Labyrinths verlassen werden würden. Das übergebene Zeichen codiert die vier möglichen Bewegungsrichtungen
  - `o` für eine Bewegung nach oben
  - `u` für eine Bewegung nach unten
  - `l` für eine Bewegung nach links
  - `r` für eine Bewegung nach rechts
- Eine Methode *toString*, welche keinen Parameter erhält aber eine Zeichenkette zurückgibt. Die zurückzugebende Zeichenkette soll das Labyrinth repräsentieren und dem eingelesenen Dateiformat entsprechen (`O` anstatt *true* und # anstatt *false*), wobei die aktuelle Position des Spielers durch ein kleines `x` dargestellt werden soll. Vergessen Sie nicht *jede* Zeile des Labyrinths innerhalb der Zeichenkette mit einen Zeilenumbruch zu versehen.