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

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

Übungsblatt 10: Objektorientierte Programmierung

Ausgabe: 16.01.2020 Abgabe: 23.01.2020

Aufgabe 1 (11 Punkte) Verwenden Sie den folgenden Freigabecode für die YAPEX-Aufgabe *Türme von Hanoi*: 5d2b807a4c04-2235

Die Türme von Hanoi sind ein mathematisches Knobelspiel, welches aus drei Stäben besteht. Auf den ersten Stab sind zu Beginn n verschieden große Scheiben übereinander angeordnet (der Größe nach aufsteigend). Ziel des Spiels ist es diese Scheiben von dem ersten Stab auf den dritten Stab zu verschieben (der zweite Stab darf dabei mit verwendet werden). Hierfür gelten allerdings die folgenden Regeln:

- es darf von einem Stab immer nur die oberste Scheibe entfernt werden, welche anschließend auf einen anderen Stab gelegt werden muss
- eine Scheibe darf nur auf einen Stab gelegt werden, wenn dort nicht bereits eine kleinere Scheibe liegt

Teil 1: Die Türme Um das Spiel programmiertechnisch umzusetzen, wird für die Stäbe (welche wir mittels des Klasse String modellieren) eine Datenstruktur ähnlich zu der eines Stacks benötigt. In der Datei `HanoiTurm.java` ist bereits der entsprechende Rahmen vorgegeben.

Implementieren Sie die folgenden Methoden in der Klasse *HanoiTurm*:

- *String top()* gibt, insofern vorhanden, die oberste Scheibe zurück, andernfalls wird `null` zurückgegeben und die Fehlermeldung `Zugriff auf leeren Stab` ausgegeben.
- *void pop()* entfernt, falls vorhanden, die oberste Scheibe, andernfalls gibt sie die Fehlermeldung `Zugriff auf leeren Stab` aus.
- *void push(String)* fügt die als Parameter übergebene Scheibe zum Stab hinzu, falls auf diesem Stab nicht bereits eine kleinere Scheibe liegt, andernfalls wird die Fehlermeldung `kleinere Scheibe vorhanden` ausgegeben. Falls die maximale Anzahl an Scheiben überschritten werden würde, soll lediglich die Fehlermeldung `maximale Anzahl an Scheiben erreicht` ausgegeben werden.
- *int size()* gibt die Anzahl der Scheiben, welche sich auf dem Stab befinden, zurück

Teil 2: Der Algorithmus Nun soll unter Verwendung der zuvor erstellten Datenstruktur ein rekursiver Algorithmus implementiert werden. Erstellen Sie dazu eine Klasse *TuermeVonHanoi*!

Ihr Programm bekommt als Kommandozeilenparameter die Anzahl der Scheiben übergeben und wendet dann den rekursiven Algorithmus an, um den Turm zu verschieben.

Jeder Stab entspricht einem Objekt der Klasse *HanoiTurm*. Ihr Algorithmus muss auf diesen Objekten arbeiten und die Scheiben mittels der in der vorherigen Aufgabe implementierten Methoden bewegen (hier erfolgt keine Ausgabe auf der Konsole!). Anschließend soll auf der Konsole ausgegeben werden, wie viele Verschiebungen durchgeführt werden mussten. Zudem soll der Turm auf dem Zielstab ausgegeben werden. Eine Scheibe der Länge 1 wird durch den String `"/\"` dargestellt, eine Scheibe der Länge 2 durch `"/\/\"` usw.

Aufgabe 2 (12 Punkte) Verwenden Sie den folgenden Freigabecode für die YAPEX-Aufgabe *Schrank mit Gegenständen*: 5d2b807abb77-f450

Bei dieser Aufgabe müssen Sie zwei Klassen erstellen: *Gegenstand* und *Schrank*. Achten Sie auf das korrekte Setzen der Zugriffsrechte!

Gegenstand Diese Klasse besitzt zwei von außen nicht zugreifbare Attribute *name* (Typ *String*) und *gewicht* (Typ *double*). Implementieren Sie die folgende Methoden:

- Konstruktor, welcher als Parameter alle notwendigen Informationen erhält, um die Attribute zu initialisieren. Sollte als Gewicht eine negative Zahl übergeben worden sein, so soll der Konstruktor das Attribut *gewicht* gleich 0 setzen.
- *getGewicht* und *getName*, welche das Gewicht bzw. die Bezeichnung zurückgeben.
- *toString*, welche einen das Objekt repräsentierenden String nach folgendem Format zurückgibt: `Name (Gewicht)`. Hat das Objekt als Bezeichnung Ördneründ als Gewicht 3, würde `Ordner (3)` als String zurückgegeben werden.

Schrank Diese Klasse verwaltet mittels eines Feldes vom Typ *Gegenstand* die einzelnen Gegenstände. Zusätzliche Informationen (aktuelle Anzahl bereits vorhandener Gegenstände, Höchstgewicht des Schrankes usw.) werden in zusätzlichen von außen nicht zugreifbaren Attributen gespeichert. Implementieren Sie folgende Methoden:

- Konstruktor, welcher als Parameter das erlaubte Maximalgewicht und die maximale Anzahl an Gepäckstücken erhält. Sie dürfen davon ausgehen, dass keine negativen Werte übergeben werden. Achten Sie darauf das Feld auch entsprechend der maximalen Anzahl an Gepäckstücken zu initialisieren!
- *packeEtwasRein* erhält als Parameter ein Objekt vom Typ *Gegenstand*, versucht es in den Schrank zu packen und gibt einen Wert vom Typ *char* zurück:
  - `v` falls der Schrank bereits voll ist.
  - `m` falls das maximale Gesamtgewicht mit dem neuen Gegenstand überschritten werden würde.
  - `e` falls der Gegenstand erfolgreich eingefügt wurde.
- *print* ohne Parameter, welche alle eingefügten Gegenstände zeilenweise auf der Konsole ausgibt.
- *print* mit dem Maximalgewicht als Parameter (*double*), welche alle eingefügten Gepäckstücke, die höchstens das Maximalgewicht aufweisen, auf der Konsole zeilenweise ausgibt.
- *istGegenstandVorhanden*, welche als Parameter einen String (die Bezeichnung des gesuchten Gepäckstückes) erhält, und `true` zurückgibt, falls ein Gepäckstück mit der Bezeichnung vorkommt, und ansonsten `false` zurückgibt.