technische universität
dortmund

Fakultät für Informatik
Lehrstühle 11, 13, 14

Jun.-Prof. Dr. Maike Buchin
Łukasz Czajka, N. Kriege, D. Kurz, M. Ludewig

Sommersemester 2019
11. Juni

DAP2 Praktikum – Blatt 12

Abgabe: 24.–28. Juni

Wichtig: Der Quellcode ist natürlich mit sinnvollen Kommentaren zu versehen. Überlegen Sie außerdem, in welchen Bereichen Invarianten gelten müssen, und überprüfen Sie diese ggf. an sinnvollen Stellen mit *Assertions* (siehe Hinweis auf Blatt 2).

Langaufgabe 12.1: Eigene Graph-Klasse (4 Punkte)

Implementieren Sie eine Klasse `Graph`, die ungerichtete Graphen modelliert. Erstellen Sie dafür zuerst die Hilfsklassen `Edge` und `Node`.

`Edge`

- Die Klasse `Edge` modelliert eine Kante zwischen zwei Knoten.
- Ihr Konstruktor soll mit zwei `Node`-Objekten aufgerufen werden.
- Diese sollen in privaten Variablen `src` und `dst` abgelegt und über passende Getter-Methoden zugänglich gemacht werden.

`Node`

- Die Klasse `Node` modelliert Knoten und enthält eine Adjazenzliste `ArrayList<Edge>`.
- Ihr Konstruktor soll mit einem Integer-Wert `id` aufgerufen werden.
- Ihre Attribute sollen als `private` deklariert und über getter-Methoden zugänglich gemacht werden.
- Sie soll die folgenden Methoden bieten:
  - `addEdge(Node dst)` soll eine Kante zwischen diesem Knoten und dem Zielknoten `dst` einfügen.
  - `equals(Object other)` soll `true` zurückgeben, wenn `other` dieselbe `id` hat wie `this`.

`Graph`

- Die Klasse `Graph` soll bei Aufruf einen leeren Graphen erzeugen. Verwenden Sie eine `ArrayList<Node>`, um die Knoten des Graphen zu verwalten.
- Sie soll die folgende Methoden bieten:
  - `boolean contains(int id)` überprüft, ob ein `Node` mit der `id` im Graphen vorhanden ist.
  - `boolean addNode(int id)` fügt einen neuen `Node` in den Graphen ein und gibt `true` zurück, wenn die `id` frei ist. Ist bereits ein Knoten mit dieser ID vorhanden, gibt die Methode lediglich `false` zurück.
  - `getNode(int id)` gibt den `Node` mit der übergebenen `id` zurück, oder `null`, wenn kein solcher `Node` existiert.
  - `addEdge(int src, int dst)` fügt eine Kante zwischen den Knoten mit den übergebenen `id`s hinzu, wenn diese vorhanden sind.
  - `static Graph fromFile(String filepath)` liest einen Graphen aus einer Textdatei aus und gibt ihn zurück (Beispielgraphen sind im Zusatzmaterial gegeben).

Testen Sie jede der Klassen zuerst einzeln mit je einer eigenen `main`-Methode.

Hinweis: Eine `toString()`-Methode, die die Attribute des Objekts als String zurückgibt, kann dabei sehr hilfreich sein.

Langaufgabe 12.2: Breitensuche (4 Punkte)

Verwenden Sie Ihre Klasse `Graph`, um den aus der Vorlesung bekannten Algorithmus zur Breitensuche auf ungerichteten Graphen zu implementieren. Gehen Sie dabei wie folgt vor:

- Schreiben Sie eine Methode `breitensuche`, der ein Graph und eine Knoten-ID `s` übergeben wird; von diesem Knoten ausgehend wird eine Breitensuche durchgeführt. Es wird also nach kürzesten Wegen von `s` zu allen anderen Knoten gesucht.
- Benutzen Sie für die Färbung der Knoten `ArrayList`s und zur Verwaltung der noch zu durchsuchenden Knoten eine `Queue` (z.B. eine `LinkedList`) oder ähnliche Klassen aus der Java-API. Nutzen Sie die Methoden `contains`, `add` bzw. `offer` und `remove` bzw. `poll`/`peek`, um die im Algorithmus beschriebenen Operationen durchzuführen.
- Implementieren Sie die `main`-Methode, die als Parameter den Pfad zu einer `.graph`-Datei und eine nicht-negative ganze Zahl erhält. Mit der Methode `Graph.fromFile` soll ein Graph aus der `.graph`-Datei ausgelesen werden, auf dem dann eine Breitensuche mit der übergebenen ganzen Zahl als Knoten-ID durchgeführt werden soll. Achten Sie insbesondere auf spezielle Rückgaben der verwendeten Methoden. Geben Sie zum Schluss den Abstand aller Knoten zum Startknoten aus, die von diesem erreicht werden können.
- **Optional**: Geben Sie zu jedem erreichbaren Knoten aus, welcher Pfad der kürzeste ist. Schreiben Sie dazu eine Hilfsmethode, die auf der `ArrayList` und dem jeweiligen Zielknoten als Parameter aufbaut. Nutzen Sie dabei aus, dass jeder Knoten nur einmal als Zielknoten einer Kante vorkommt und überlegen Sie, warum das so ist.