Übung 09: Listen, Sets und Maps in der Datenverarbeitung

In dieser Übung verwenden Sie die Interfaces `List<T>`, `Set<T>`, `Map<K, V>` sowie das innere Interface `Map.Entry<K, V>`, welche alle in `java.util` eingeordnet sind. Mithilfe von `Iterator<T>`, welcher von Containerklassen bereit gestellt wird, die `Iterable<T>` implementieren (z.B. `LinkedList<T>`, `TreeSet<T>`), berechnen Sie Statistiken über Tweets, am Beispiel einer prominenten Person.

Ziel dieser Aufgabe

Zunächst werden nach und nach Tweets (Strings) indiziert, um auszuzählen, welches Wort wie oft vorkommt. Aufbauend auf dieser Statistik werden nun Iteratoren erstellt, um über

- das gesamte Vokabular (alphabetisch),
- die häufigsten Hashtags (absteigend, #...),
- die häufigsten Wörter (absteigend, quasi Buzzwords) zu iterieren.

In einem letzten Schritt wird ein Iterator erstellt, welcher über alle Tweets iteriert und absteigend nach "Buzzwordgewicht" sortiert.

Von technischer Seite her sind das Interface `TweetSammlung` sowie die Factorymethode `TweetSammlung.create` zu implementieren.

Hinweise

- In der offiziellen Java-Dokumentation finden Sie für jedes der Interfaces (z.B. `List<T>`) eine Auflistung der implementierenden Klassen (z.B. `LinkedList<T>` und `ArrayList<T>`, uvm.).
- Es bildet ungemein, sich auch die Javadoc zu den anderen Methoden und Interfaces durchzulesen, die verwendet werden.
- In dieser Übung müssen Sie keinen (!) eigenen Iterator implementieren; vielmehr legen Sie Datenstrukturen an, welche Sie sortieren und filtern, und beim Ergebnis `.iterator()` aufrufen.

Aufgabe 1: Klasse und Factorymethode

Erstellen Sie eine neue Klasse, welche `TweetSammlung` implementiert, und ändern Sie die statische Factorymethode `TweetSammlung.create` so, dass eine neue Instanz dieser Klasse zurückgegeben wird.

Aufgabe 2: Tweets indizieren

Implementieren Sie die Methode `ingest(String tweet)`. Diese Methode nimmt einen String entgegen, welcher mit der statischen Methode `TweetSammlung.tokenize(String)` in eine `List<String>` umgewandelt werden kann. Der originale Tweet soll in einer Liste (`List<String>`) abgelegt werden. Für jedes Wort soll ein Zähler unterhalten werden, wie oft ein Wort bereits insgesamt vorgekommen ist.

Diese Methoden werden Sie außerdem brauchen:

- `List<T>.add(T e)`: hängt ein Element an eine Liste an
- `Map<K, V>.get(K key)`: gibt den Wert für einen Schlüssel zurück, oder `null`
- `Map<K, V>.containsKey(K key)`: gibt an, ob es für einen Schlüssel einen Wert gibt
- `Map<K, V>.put(K key, V val)`: setzt den Wert (`val`) für einen Schlüssel (`key`)

Aufgabe 3: Einfache Iteratoren

Implementieren Sie die folgenden Methoden:

- `Iterator<String> vocabIterator()`: Gibt einen Iterator über alle Wörter zurück, alphabetisch sortiert. Hinweis: Verwenden Sie `Map.keySet()` und lesen Sie die Dokumentation dazu.
- `Iterator<String> topHashTags()`: Gibt einen Iterator über alle Hash-Tags (also Wörter die mit # beginnen) zurück, absteigend sortiert nach Häufigkeit. Hinweis: Je nach Implementierung benötigen Sie eine "Hilfsliste"; um Wörter und deren Häufigkeit in einer solchen Liste zu speichern verwenden Sie z.B. die Klasse `org.apache.commons.lang3.tuple.Pair` (via `Factorymethode Pair.of()`).
- `Iterator<String> topWords()`: Gibt einen Iterator über alle die Wörter zurück, deren erster Buchstabe alphabetisch ist (siehe `Character.isAlphabetic`), ebenfalls absteigend nach Wichtigkeit.

Diese Methoden werden Sie außerdem brauchen:

- `List<T>.sort(Comparator<T> c)`: Sortiert eine Liste mit gegebenem Comparator.
- Das Interface `Map.Entry<K, V>`, welches einen Schlüssel `.getKey()` und einen Wert `.getValue()` enthält; siehe auch `Map.entrySet()`.
- `Pair.getLeft()` und `Pair.getRight`: Gibt das linke bzw. rechte Element eines Datenpaares zurück; `Pair<K, V>` implementiert `Map.Entry<K, V>`.

Aufgabe 4: Beste Tweets

Implementieren Sie die Methode `Iterator<Pair<String, Integer>> topTweets()`, welche einen Iterator zurückgibt, der über Paare von Tweets und "Buzzwordlastigkeit" (`Pair<String, Integer>`)iteriert. Die Paare sollen absteigend nach Buzzwordlastigkeit sortiert sein, die Buzzwordlastigkeit eines Tweets ist die Summe der Häufigkeiten der Wörter.

Hinweis: Je nach Implementierung müssen Sie hier wieder eine Hilfsliste erstellen, die zunächst Paare von Tweet und Buzzwordlastigkeit (also `Pair<String, Integer>`) enthält, welche dann entsprechend sortiert werden können.

Aufgabe 5: Stopwörter

Wie Sie den Testausgaben entnehmen können, treten Allerweltswörter wie "so" oder "and" in den Vordergrund, obwohl diese offensichtlich unwichtig sind. Implementieren Sie die Methode `setStopwords(File)`, welche mit einem `java.util.Scanner` zu ignorierende Wörter aus einer Datei liest (`siehe java.io.FileReader`). Lesen Sie dazu Wort für Wort in ein separates `Set<String>`, und passen Sie anschließend die Methoden `ingest` und `topTweets` an, sodass im `Stopwordset` enthaltene Wörter ignoriert werden. Verwenden Sie dazu die Methode `Set<T>.contains(T e)`.