Aufgabenblatt 4 – 13.04.2021

Programmieren II: Fortgeschrittene (Java) Abgabe bis 20.04.2021

Aufgabe 1) Supertypen **Punkte: 3**

In den mitgelieferten Dateien finden Sie (vereinfachte) Varianten der aus dem letzten Blatt bekannten Klassen für die Darstellung von Sätzen.

- `Sentence` ist die bekannte Darstellung von Sätzen mittels einer Liste aus Strings
- `TaggedSentence` speichert zusätzlich Tags in einer separaten Liste
- `TaggableSentence` ist eine Subklasse von `TaggedSentence`, die es erlaubt, die Tags zu verändern.
- `DependencySentence` speichert einen Depenzbaum, aber keine Tags.
- `TokenSentence` speichert sowohl Tags, als auch Dependenzinformationen in `Token` Objekten.

Daneben sind in `SentenceUtils` einige Methoden implementiert, die auf den verschiedenen Satz-Klassen arbeiten. Da die Methoden implementiert wurden, bevor `TokenSentence` existierte, kann keine der Methoden mit `TokenSentence` arbeiten, obwohl `TokenSentence` eigentlich alle benötigten Funktionalitäten zur Verfügung stellt. Dies wollen wir ändern.

- Sorgen Sie dafür, dass die Methoden in `SentenceUtils` *zusätzlich* auch mit `TokenSentence` verwendet werden können.
- Sie dürfen zusätzliche Typen (z.B. Interfaces) einführen, Klassen umbenennen und die Typen von Parametern verändern. Die Implementierung der Klassen oder der Methoden an sich soll allerdings nicht verändert werden.

Aufgabe 2) Alternative Tokens **Punkte: 5**

Die ebenfalls aus dem letzten Blatt bekannte und im Begleitmaterial mitgelieferte Klasse `Token` speichert jedes Token isoliert als String. Dabei gehen bestimmte Informationen über den Ausgangstext verloren, beispielsweise wenn Tokens durch mehr als ein Leerzeichen getrennt werden. Wir wollen deshalb eine alternative Tokenrepräsentation implementieren. Die Idee ist, jedes Token mit seiner Anfangsposition und seiner Länge im Originalstring darzustellen. Beispielsweise würden wir den Text `This␣is␣a␣␣test` durch vier Token darstellen: `(0, 4)`, `(5, 2)`, `(9, 1)`, `(11, 4)`. Hierbei ist die erste Zahl die Position (in Buchstaben) an der das Token anfängt und die zweite Zahl die Länge

des Tokens.

- Schreiben Sie eine Klasse `PointerToken`. Die Klasse soll eine Referenz auf einen String `original` speichern, der als Originalstring dient. Daneben soll die Klasse eine Instanzvariablen `start` enthalten, die den Anfang des Tokens in `original` enthält, sowie eine Variable `length`, die dessen Länge speichert. Alle drei Variablen sollen in einem Konstruktor initialisiert werden. (1 Punkt)

- Schreiben Sie eine Methode `getSurface()`, die den Text des Tokens zurückgibt (1 Punkt)

- Wir wollen mit der neuen Klasse dieselben Zusatzinformationen speichern, wie in `Token`. `PointerToken` soll also genauso wie `Token` POS-tags, Dependenzen usw. speichern können. `PointerToken` soll anschließend überall da eingesetzt werden können, wo `Token` verwendet werden kann. Setzen Sie dies in Ihrem Code um. Sie dürfen dafür sowohl neue Klassen und Interfaces einführen, als auch Code verschieben. Achten Sie darauf, Code Duplikation zu minimieren. (3 Punkte)

Aufgabe 3) Token Highlighter **Punkte: 2**

Im letzten Schritt wollen wir unsere eigene Funktionaltität zu `SentenceUtils` hinzufügen. Wir wollen eine Methode schreiben, die die Oberflächenformen (`surface`) eines Satzes durch Leerzeichen getrennt in einen String zusammensetzt. Jedes Token, das einen bestimmten POS-Tag hat, soll dabei in Großbuchstaben angehängt und durch zwei Asterisk (`*`) umgeben werden.

- Schreiben Sie eine Methode `static String highlightSentence(T sentence, String tag)`, die alle Tokens in `sentence` durch Leerzeichen getrennt ausgibt. Alle Tokens, die den POS-Tag `tag` haben, sollen dabei komplett Großbuchstaben ausgegeben werden. (1.5 Punkte)

- Wählen Sie den Typ `T` so, dass er im Rahmen Ihrer Typenhierarchie mit allen Klassen arbeiten kann, die POS-tags speichern. (0.5 Punkte)

Abgabemodalitäten
Reichen Sie ihre Lösung pünktlich bis 23:59 am genannten Abgabetermin per auf Moodle ein ein.
Gruppenabgaben von bis zu maximal 3 Personen sind erlaubt.
Fügen Sie zur Abgabe ein Dokument "Gruppenmitglieder.txt" hinzu, das die Vor- und Nachnamen aller Gruppenmitgliedern auflistet.
Reichen Sie pro Gruppe nur eine Abgabe ein.| Bei offensichtlichem Abschreiben werden alle beteiligten Abgaben mit 0 Punkten bewertet.