Fachbereich Informatik und Kommunikation
Prof. Dr. Marcel Luis

Westfälische Hochschule

6. Juni 2012

Aufgabenblatt 8 zu Objektorientierte Programmierung

Aufgabe 8.1

In dieser Aufgabe geht es

- *allgemein* um die wortweise Verarbeitung von Texten aus beliebigen zeichenorientierten Datenquellen und
- *speziell* um die Indexierung von Texten.

Realisieren Sie dazu im Paket `textverarbeitung` eine Klasse `Textverarbeiter`. Aufgabe eines Objekts dieser Klasse ist es, den gesamten Text einer zeichenorientierten Datenquelle einzulesen und die einzelnen Wörter einem Wortverarbeiter zur weiteren Verarbeitung zu übergeben.

Implementieren Sie in der Klasse `Textverarbeiter` folgende Methoden:

- Einen Konstruktor `Textverarbeiter(Wortverarbeiter)`. Durch den Parameter wird dem Textverarbeiter der Wortverarbeiter übergeben, der die Verarbeitung der einzelnen Wörter übernimmt.
- Eine Instanzmethode `void verarbeite(Reader) throws IOException` zur Verarbeitung des Texts aus der übergebenen Datenquelle. Zeichen zur Worttrennung sind `.,:;!?-()` sowie das Leerzeichen.

Realisieren Sie im Paket `textverarbeitung` außerdem eine Schnittstelle `Wortverarbeiter` mit folgenden Methoden:

- Eine Instanzmethode `void verarbeite(String)` zur Verarbeitung eines Worts.
- Eine Instanzmethode `void verarbeiteZeilenende()`, durch die dem Wortverarbeiter mitgeteilt wird, dass das Ende einer Textzeile erreicht wurde.

Ein Textverarbeiter kann nun zusammen mit geeigneten Wortverarbeitern beliebige Operationen auf den Wörtern eines Texts ausführen. *Eine* mögliche Operation ist es, einen Wortindex für einen Text zu erstellen. Der Wortindex eines Texts besteht aus den Wörtern des Texts (bestimmte Ausschlusswörter sollen ausgenommen werden) und der Information, in welchen Zeilen des Texts die einzelnen Wörter vorkommen. Der Index enthält kein Wort doppelt.

Realisieren Sie im Paket `textverarbeitung` eine Klasse `Indexierer`, die mit einem Textverarbeiter verwendet werden kann, mit folgenden Methoden:

- Einen Konstruktor `Indexierer(Collection<String>)`, durch den ein Indexierer erzeugt wird. Der Parameter gibt die Ausschlusswörter an.

- Eine Instanzmethode `List<String> gibWoerter()`, die alle Wörter des Index in alphabetisch sortierter Reihenfolge liefert. Es soll die „gewöhnliche“ Ordnung auf Strings verwendet werden.

- Eine Instanzmethode `String gibZeilennummern(String)`, die in Form einer Zeichenkette alle Zeilennummern zu einem Wort liefert. Die Zeichenkette enthält die Zeilennummern in aufsteigender Reihenfolge, jeweils durch Komma und Leerzeichen getrennt. Keine Zeilennummer erscheint doppelt.

  Beispiel: Kommt ein Wort in einem Text in den Zeilen 1, 3, 4 und 24 vor und indexiert man diesen Text, dann liefert diese Methode für das Wort die Zeichenkette `"1, 3, 4, 24"`.

  Ist das übergebene Wort nicht im Index enthalten, liefert die Methode die leere Zeichenkette.

Schreiben Sie außerdem zwei Testklassen `TextverarbeiterTest` und `IndexiererTest`. Überlegen Sie sich dafür selbst sinnvolle Testmuster. Selbstverständlich ist es ratsam, mit den Testklassen zu beginnen.

Beispiel für den Index eines Texts. Die Ausschlusswörter des Indexierers seien: in, und

Text	Index
In Ulm, um Ulm und um Ulm herum.	Ulm: 1, 2 In: 1 um: 1, 2 herum: 2

Hinweise

- In dieser Aufgabe gibt es mehrere „Stellen“, an denen *viele Objekte* eines Typs verwaltet werden müssen: Ein Indexierer verwaltet *viele Wörter*; einem Wort sind *viele Zeilennummern* zugeordnet; ein Indexierer besitzt *viele Ausschlusswörter*. Überlegen Sie, wie Sie hier die Collection-Klassen des JDK sinnvoll einsetzen können. Lassen Sie sich vor allem nicht von den vorgegebenen Typen der Methoden in eine unglückliche Richtung leiten.

- Eine Anwendung der Klassen `Indexierer` kann darin bestehen, größere Texte aus Dateien zu indexieren. Für den Test ist es jedoch sinnvoll, mit internen Datenquellen zu arbeiten.

In der Vorlesung haben wir besprochen, welche konkrete `Reader`-Klasse dafür zweckmäßig verwendet wird.

- Für den Test der Klasse `Textverarbeiter` benötigen Sie einen Wortverarbeiter. Welcher ist hierfür sinnvoll?

- Denken Sie an die ausreichende Dokumentation und Kommentierung Ihrer Lösung. Beachten Sie die unterschiedliche Bedeutung der *externen Dokumentation* `/** ... */` *vor* einer Klasse oder Methode und des *Implementierungskommentars* `/* ... */` *innerhalb* einer Methode. Die externe Dokumentation sagt, *was* eine Klasse oder Methode leistet, der Implementierungskommentar hilft zu verstehen, *wie* es gemacht wird. Verwenden Sie Implementierungskommentare vor allem, um den Berechnungsablauf verständlich zu machen.

- Erzeugen Sie die HTML-Dokumentation Ihrer Klasse und überzeugen Sie sich, ob Ihre externe Dokumentation sinnvoll und ohne Kenntnis des Quellcodes der Klasse hilfreich ist.

- Erstellen Sie je ein `zip`-Archiv des Ordners `textverarbeitung` unter `src` und des gleichnamigen Ordners unter `test` und laden Sie beide zu Moodle hoch.