Programmieren 2 (INF)
Übungsblatt 6

Dieses Übungsblatt beschäftigt sich mit dem Einsatz von Standardcontainern aus dem Java Collection-Framework. Als Anwendungsbeispiel dient die Analyse einer Textdatei. Es soll die Häufigkeit der Vorkommen einzelner Wörter in einem Text ermittelt und in lesbarer Form ausgegeben werden. Die Teile des Programms, die sich mit dem Lesen aus Dateien beschäftigen, werden dabei vorgegeben. Es ist hier nicht das primäre Lernziel, den Umgang mit Strömen und Dateien zu erlernen (kommt später)!

Aufgabe 1

Legen Sie ein Verzeichnis `data` im Wurzelverzeichnis Ihres Java-Projekts an. Speichern Sie darin einen Liedtext Ihrer Wahl als ASCII-Datei unter dem Namen `Song.txt` ab.
(Pfad könnte z.B. sein: `workspace/Programming2/data/Song.txt`)

Aufgabe 2

Gegeben Sei das folgende Programm:

```
public class WordFrequencyCounter {
	public void analyzeText(File textFile) {
		try {
			BufferedReader in =
				new BufferedReader(new FileReader(textFile));
			// Iterate through each line of the file
			while (true) {
				String currLine = in.readLine();
				if (currLine == null)
					break;
				analyzeLine(currLine);
			}
		} catch (IOException ex) {
			System.out.println(
				"Error occurred while reading from "
				+ textFile.getAbsolutePath() + ":");
			System.out.println(ex);
		}
	}
	private void analyzeLine(String line) {
		System.out.println(line);
	}
	public void printResults() {
	}

	public static void main(String[] args) {
		WordFrequencyCounter counter = new WordFrequencyCounter();
		counter.analyzeText(new File("data/Song.txt"));
		counter.printResults();
	}
}
```

Versuchen Sie, zu verstehen, was das Programm macht und wie. Legen Sie für dieses Übungsblatt ein neues Paket an, kopieren Sie das Klassenfragment hinein und versuchen Sie, die `main`-Methode zum Laufen zu bringen.

Aufgabe 3

Jetzt wird’s interessant. Das Programm soll nun so erweitert werden, dass es in der Textdatei analysieren kann, wie viele Zeilen es gibt, wie viele Wörter und welches Wort wie oft vorkommt. Dabei sollen die Wörter nach Häufigkeit sortiert ausgegeben werden. Fügen Sie geeignete Attribute und Anweisungen in das Programmfragment ein. Sie werden vermutlich auch eine weitere Hilfsklasse benötigen. Auf der nächsten Seite finden Sie bei Bedarf ein paar Tipps zur Implementierung.

Der folgende Text zeigt das Analyseergebnis der Musterlösung aus einem Beispiellauf. Erkennen Sie den Song?

```
Line Count: 31
Word Count: 186
Frequency of occurrence of each word:
         i :  10  beautiful :   9     you're :   7       that :   6       with :   6
         a :   5         be :   5       face :   5         my :   5        she :   5
      it's :   4         on :   4        you :   4     'cause :   3        and :   3
       but :   3      don't :   3       i'll :   3         is :   3      never :   3
       saw :   3        the :   3         to :   3       true :   3         an :   2
     angel :   2  brilliant :   2    crowded :   2         do :   2        her :   2
        in :   2       know :   2       life :   2      place :   2        see :   2
       was :   2         we :   2       what :   2       will :   2       your :   2
    you´re :   2      again :   1    another :   1         as :   1         at :   1
        by :   1     caught :   1      could :   1        end :   1        eye :   1
      from :   1    fucking :   1        got :   1       high :   1        i'm :   1
      i've :   1       last :   1       lose :   1       love :   1        man :   1
        me :   1     moment :   1       must :   1         no :   1         of :   1
      plan :   1       pure :   1     shared :   1     should :   1      sleep :   1
     smile :   1     smiled :   1     subway :   1       sure :   1      there :   1
     think :   1    thought :   1       till :   1       time :   1      truth :   1
        up :   1     walked :   1       when :   1      won't :   1       yeah :   1
```

Tipps

- Die Zerlegung einer Textzeile in seine Wörter kann mit Hilfe der Klasse `java.util.StringTokenizer` durchgeführt werden. Lesen Sie bei Bedarf in der Dokumentation nach.
- Es macht wenig Sinn, nach Groß- und Kleinschreibung zu differenzieren. Die Klasse `String` enthält dafür eine praktische Methode.
- Welche Datenstruktur eignet sich besonders für die Häufigkeitszählung? Wird im Kern nicht eine Abbildung benötigt? Dafür lohnt es sich, ein Attribut anzulegen!
- Bei der Ergebnisdarstellung kann man sich die Sortierfunktionen der Klasse `Collections` zu Nutze machen.
  Idee: Einträge aus der Abbildung in eine Liste füllen und dann sortieren.
  Warum brauchen wir dafür einen speziellen `Comparator`?
- Schließlich gibt es zur formatierten Ausgabe noch einen Tipp…
  `System.out.printf("%10s : %3d ", <word>, <number>);`

Jetzt alles klar?