Prof. Dr. Harald Brandenburg
Hochschule für Technik und Wirtschaft (HTW)
Fachbereich 4 (Wirtschaftswissenschaften II)
Wilhelminenhofstraße 75 A
12459 Berlin (Oberschöneweide)
Raum WH C 605

Fon: (030) 50 19 - 23 17
Fax: (030) 50 19 - 26 71
h.brandenburg@htw-berlin.de

Freitag, 11. Juni 2010

Programmierung 2

SS 2010

Aufgabe 7:	**Gruppe 1:** 24.06.10	**Gruppe 2:** 01.07.10

Schreiben und dokumentieren Sie ein Programm, mit dem festgestellt werden kann, welche Wörter wie häufig in einer Textdatei vorkommen. Es soll (mindestens) folgenden Leistungsumfang haben:

- Der Name der zu untersuchenden Textdatei soll dem Programm beim Aufruf als Parameter übergeben werden. Bei zu wenig oder zu vielen Parametern soll eine Anleitung ausgegeben werden.

- Die Textdatei soll geöffnet und gelesen werden. Das Programm soll sich aber auch dann vernünftig verhalten, wenn die Datei nicht vorhanden ist oder nicht geöffnet werden kann.

- Das Programm soll aus dem gelesenen Text einzelne Wörter extrahieren und deren Häufigkeit zählen. Dazu soll es im Hauptspeicher eine „Liste“ aufbauen, die jedes im Text vorkommende Wort mit der entsprechenden Häufigkeit enthält.

- Es soll aus folgendem Menü gewählt werden können:

```
(1)   Woerter aufsteigend alphabetisch sortiert
(2)   Woerter mit Haeufigkeit von min bis max aufsteigend alphabetisch
      sortiert
(3)   Woerter absteigend alphabetisch sortiert
(4)   Woerter mit Haeufigkeit von min bis max absteigend alphabetisch
      sortiert
(5)   Woerter absteigend nach Haeufigkeit sortiert
(6)   Woerter mit Haeufigkeit von min bis max absteigend nach Haeufigkeit
      sortiert
(7)   Woerter aufsteigend nach Haeufigkeit sortiert
(8)   Woerter mit Haeufigkeit von min bis max aufsteigend nach Haeufigkeit
      sortiert
(9)   Anzahl Woerter feststellen
(10)  Woerterliste speichern
(11)  Neue Woerterliste erzeugen
(12)  Textdatei anzeigen
(13)  Programm beenden
```

- Bei Wahl von `(1)` soll auf dem Bildschirm eine Aufstellung folgender Art erscheinen, wobei die Wörter aufsteigend in alphabetischer Reihenfolge ausgegeben werden:

```
Ergebnis der Analyse der Datei dateiname.txt:

wort1        5
wort2       17
wort3        1
......     ...
```

- Bei Wahl von **(2)** soll die Aufstellung nur solche Wörter enthalten, deren Häufigkeit zwischen **min** und **max** liegt, wobei **min** und **max** zuvor erfasst wurden.

- Bei Wahl von **(3)** bis **(8)** sollen die Wörter entsprechend der gewünschten Sortierart und Häufigkeit ausgegeben werden. Bei gleicher Häufigkeit sollen sie zusätzlich alphabetisch sortiert sein.

- Bei Wahl von **(9)** soll auf dem Bildschirm ausgegeben werden, wie viel Wörter die Wörterliste enthhält:

  ```
  Die Liste enthaelt 64 Woerter.
  ```

- Bei Wahl von **(10)** soll die aktuelle Wörterliste[1] in eine Textdatei geschrieben werden, deren Name frei gewählt werden kann. Es muss dabei sichergestellt werden, dass keine Datei unbeabsichtigt überschrieben wird.

- Bei Wahl von **(11)** soll der Name einer Textdatei erfasst werden, die anschließend analysiert werden kann. Das Programm soll sich auch dann vernünftig verhalten, wenn die Datei nicht vorhanden ist oder nicht geöffnet werden kann.

- Bei Wahl von **(12)** soll eine beliebige Textdatei gelesen und deren Inhalt auf dem Bildschirm ausgegeben werden. Der Name der Textdatei soll über die Tastatur erfasst werden.

[Hinweise:

- Textdateien können zum Beispiel mit einem **FileReader** (in Kombination mit einem **BufferedReader**) gelesen werden. Seit der Java-Version 1.5 steht hierfür aber auch die Klasse **Scanner** zur Verfügung. Sie können mit einem **FileWriter** gespeichert werden.

- Zum Extrahieren von Wörtern aus Zeichenketten kann die Klasse **StringTokenizer** eingesetzt werden. Als einfache Alternative steht auch die Methode **String.split(...)** zur Verfügung. Komplizierter ist die Nutzung eines **StreamTokenizer**.

- Es bleibt Ihnen überlassen, zu definieren, was ein "Wort" ist. Gute Lösungen berücksichtigen auch Worttrennungen, Bindestriche, Hochkommata, Schrägstriche, Unterstriche, etc., ebenso Ziffern oder sonstige Zeichen. Das Programm braucht nur für ASCII-Texte zu funktionieren. Texte, die mit modernen Textverarbeitungsprogrammen erstellt wurden, werden in einem eigenen Format gespeichert, das nicht so einfach zu untersuchen ist!

- Die Wörterliste sollten Sie in einer **LinkedList** (oder **ArrayList**) speichern, die Sie mit Hilfe eines **ListIterator** durchlaufen. Für Teile der Aufgabe können Sie aber auch einfach eine **HashMap** bzw. eine **TreeMap** benutzen, die Sie bei Bedarf (z.B. zum Sortieren) in eine **LinkedList** verwandeln.

- Laden Sie bitte das Buch *Kritik der reinen Vernunft, Zweite hin und wieder verbesserte Auflage (1787)* von Immanuel Kant herunter, das Sie unter der Adresse

  http://www.gutenberg.org/etext/6343

  finden (Plain text). Sie sollen Ihre Lösung anhand dieses Textes präsentieren.

]

1 Das ist das jeweilige Ergebnis der Operationen **(1)** – **(8)**.