Übungen zu Praktische Informatik: Datenstrukturen **SS 2006**

Übung 1

Name: _____ Tutor: _____

Matrikelnummer: _____ Punkte: _____

Gruppe: _____ Abgabe am: Di, 21.03. 2006 12:00

Wortliste (Punkte: maximal 24; Teilaufgaben: a=11, b=11, c=2)

Implementieren Sie eine einfach verkettete, lineare Liste, die die Häufigkeit der eingefügten Wörter zählt und statistische Auswertungen ermöglicht. Es soll nicht zwischen Groß- und Kleinschreibung unterschieden werden.

Achten Sie darauf, dass nur die angegebenen Methoden *public* sind, alles andere soll *private* deklariert sein (oder zumindest gut begründet wenn nicht).

```
public class WordList0 {
     public WordList0() { … }
     public void insert(String word) { … }
     // Get frequency of word
     public int getFrequency(String word) { … }
     // Number of different words
     public int nOfDifferentWords() { … }
     // Mean frequency of equal words
     public double meanWordCount() { … }
     // Get all words with specified frequency
     public String[] getWords(int frequency) { … }
     // Get all words starting with the specified prefix
     public String[] getWordsStartingWith(String prefix) { … }
     public void printList() { … }
}
```

Beispiel: Einlesen von "To be or not to be" soll beim Aufruf der Methode *printList* in etwa folgende (Test-)Ausgabe liefern (die Reihenfolge der Wörter ist nicht relevant):

```
1 NOT (1 times)
2 BE (2 times)
3 OR (1 times)
4 TO (2 times)
```

Die Methode *nOfDifferentWords* liefert in diesem Beispiel das Ergebnis 4, *meanWordCount* liefert 1.5 (versuchen Sie bei der Implementierung ohne Schleifen auszukommen).

a) Implementieren Sie eine „einfache“ Wortliste (in der Klasse *WordList0*) laut obiger Spezifikation.

b) Implementieren Sie *WordList1 als* Erweiterung von *WordList0,* als eine „selbstorganisierende“ Wortliste. Implementieren Sie dazu die Methoden:

- `public WordNode find(String word):` zum Suchen eines Elements mit gegebenem Wort, und Verschieben an den Anfang. Wird das Element nicht gefunden, soll null zurückgegeben werden.
- `public WordNode remove(String word):` zum Löschen *eines* Elements mit gegebenem Wort.
- passen Sie `public void insert (String word)` an, damit das zuletzt eingefügte Element immer am Anfang steht.

Hinweis: Sie können in *remove* und *insert* die Methode *find* verwenden (Sonderfälle beachten!)

c) Testen Sie Funktionalität und Effizienz Ihrer Lösung