Übung 03: Binärbaum / Rot-Schwarz-Baum

Abgabetermin: 24.04.2007 12:00

Name: _____ **Matrikelnummer**: _____

Gruppe: ❒ G1 (Dhungana) ❒ G2 (Wolfinger) ❒ G3 (Wolfinger)

Aufgabe	Punkte	gelöst	abzugeben schriftlich	abzugeben elektronisch	Tutor	Pkte
Aufgabe 03.1	20	❑	Java-Programm Testfälle und Ergebnisse Effizienzvergleich Überlegungen zum Balancieren	Java-Programm		
Aufgabe 03.2	4	❑	Rot-Schwarz-Bäume	-		

Aufgabe 03.1: Wortliste Binärbaum (20 Punkte, Teilaufgaben: a=10, b=3, c=7)

a) Um die Wortliste aus Übung 2 zu beschleunigen soll sie in der Klasse `TreeWordList` als Binärbaum implementiert werden.

```
class TreeWordList
  implements WordList {
  private WordNode2 root;
  public TreeWordList () { … }
  …
}
```

```
class WordNode2 {
  int count;  // frequency
  String word;
  WordNode2 left, right;
  public WordNode2(…) {… }
}
```

Die Klasse `TreeWordList` soll das folgende Interface `WordList` aus Übung 2 implementieren:

```
public interface WordList {
     // Insert a word
     void insert(String word);
     // Remove a word
     void remove(String word);
     // Get frequency of word
     int getFrequency(String word);
     // Number of different words
     int nOfDifferentWords();
     // Mean frequency of equal words
     double meanWordCount();
     // Get all words with specified frequency
     String[] getWords(int frequency);
     // Get all words starting with the specified prefix
     String[] getWordsStartingWith(String prefix);
     void print();
     // Creates a copy
     WordList clone();
     // Merges two lists
     void merge(WordList list);
     // Gets an iterator
     Iterator iterator();
}
```

b) Testen Sie die Effizienz Ihrer Lösung (mit Hilfe des Programms *TestEfficiency.java* (als Download auf der Übungsseite im Internet). Welche Methoden werden wie viel schneller ausgeführt als bei der Übung 2? Gibt es auch Methoden, die gleich schnell/langsamer ausgeführt werden? Überlegen Sie (schriftlich!) die Gründe für die jeweiligen Änderungen der Effizienz der einzelnen Methoden. Wodurch ergeben sich die Unterschiede?

c) Implementieren Sie das Balancieren des Binärbaumes. Das Balancieren soll dabei im Hintergrund erfolgen (d.h. der Benutzer der Wortliste soll sich darum nicht kümmern müssen). Überlegen Sie (schriftlich?): Von wo aus soll die Methode *balance* aufgerufen werden? Wie oft/unter welchen Kriterien müsste *balance* aufgerufen werden, damit es Vorteile bringt? Testen Sie die Wortliste mit und ohne balancieren. Werden die Erwartungen erfüllt?

Implementierungshinweis:

- Die Methoden *clone(), merge()* und *iterator()* müssen nicht implementiert werden. Implementieren Sie diese Methoden mit Hilfe einer *UnsupportedOperationExeption()* wie folgt:

```
public WordList clone() {
  throw new UnsupportedOperationException();
}
public void merge(WordList list) {
  throw new UnsupportedOperationException();
}
public Iterator iterator() {
  throw new UnsupportedOperationException();
}
```

- Testen Sie Ihre Implementierung mit dem JUnit-Test *TestWordList2.java* (als Download auf der Übungsseite im Internet). JUnit 4.0 bekommen Sie unter http://www.junit.org.
- Ihre Implementierung der Klassen *WordNode*, *SortedWordList* aus Übung 2 wird in dieser Übung zum Vergleich der Effizienz benötigt. Übernehmen Sie die Klassen aus Ihrer Ausarbeitung für Übung 2.

Abzugeben ist:

- Das Java-Programm
- Testfälle und die Ergebnisse
- schriftliche Überlegungen zur Effizienz
- schriftliche Überlegungen zum Balancieren im Hintergrund

Aufgabe 03.2: Rot-Schwarz-Baum zeichnen (zur Präsentation an der Tafel vorbereiten)

Geben Sie Rot-Schwarz-Bäume an (mit allen Zwischenschritten), die beim Einfügen der Buchstaben CATCHERINTHERYE entstehen.