Technische Universität Berlin **SS99**
Fachbereich Mathematik
Sekr. MA 6-1 / Frau Marcus

Dr. Michael Joswig Olaf Jahn
Matthias Müller, André Opitz

3. Programmieraufgabe

Abnahme: spätestens 01.06.99 zur Rechnervorrangzeit

Schreibt ein Java-Programm, das beliebige Dateien mit Hilfe von Huffman-Codes codieren und wieder decodieren kann. Es soll also jeweils eine Datei lesen und in eine andere die komprimierte bzw. dekomprimierte Version schreiben.

Was die Gestaltung Eures Programms angeht, seid Ihr (fast) völlig frei, wir geben keine Interfaces als Zwang vor. Es ist zum Beispiel Euch auch überlassen, ob Ihr eine Fensteroberfläche vorseht oder nicht, ob Ihr zwei verschiedene `main(String)` zum Codieren und Decodieren haben wollt, oder beides über einen Einsprungspunkt macht.

Ein paar Dinge beschränken die grenzenlose Freiheit allerdings:

- Der Benutzer soll jeweils die Namen der Quell- und der Zieldatei vorgeben können.
- Durch Eurer Programm geschriebene komprimierte Dateien sollen auch von Euch wieder dekomprimierbar sein. Die dekomprimierte Datei muß *exakt* mit der Originaldatei übereinstimmen (Test mit Unix-Kommando `cmp`).
- Sämtliche zum Decodieren notwendigen Informationen müssen in der komprimierten Datei vorhanden sein, also auch Informationen über den Codebaum.
- Ihr dürft keine Datei jemals komplett in den Speicher lesen. Das bedeutet z. B., daß Ihr zum Komprimieren die Originaldatei zweimal lesen müßt, einmal zum Zählen der Häufigkeiten und einmal zum eigentlichen Codieren.
- Die zur Erzeugung der Huffman-Codes notwendige PriorityQueue soll als binärer Heap implementiert werden.
- Ab einer angemessenen Dateigröße (und bei entsprechend komprimierbaren Daten) soll eine komprimierte Datei auch spürbar kürzer als das Original sein. Achtet darauf, in der komprimierten Datei möglichst sparsam mit Platz umzugehen.
- Es müssen sich beliebige Dateien komprimieren lassen, auch solche, die keinen reinen ASCII-Text enthalten. Daher ist es notwendig, daß Ihr Bytes statt Characters einlest und schreibt (mit den Standardklassen `BufferedInputStream` und `FileInputStream` etc.) Den `StringTokenizer` könnt Ihr bei dieser Aufgabe vergessen.
- Dateien, die nur aus der Wiederholung des gleichen Bytes bestehen, müssen komprimiert werden können. Der Codebaum hat in diesem Fall eine besondere Gestalt.

Auf unseren WWW-Seiten stellen wir Euch u. a. Klassen zum Lesen und Schreiben von Bits, die Ihr verwenden könnt – aber nicht müßt.

Tip: Bei dieser Programmieraufgabe ist es wichtiger denn je, daß Ihr ein *klares* Konzept habt und einen ebensolchen Kopf behaltet. Darum überlegt Euch vor dem Tippen, welche Funktionalität Ihr benötigt und in welchen Klassen sie implementiert werden soll. Zerlegt Eurer Problem in *überschaubare* Einzelteile, die entsprechend gekapselt sind und sinnvollerweise separat getestet werden sollten.

Mit Kuddel-Muddel-Java werdet Ihr wahrscheinlich Schiffbruch erleiden. Daher laßt Euch bitte bei Problemen schon frühzeitig von einem von uns beraten.