DeleteLines, Pipe

1 DeleteLines

Schreibe ein Programm, welches über Kommandozeilenparameter gesteuert wird. Beispielaufruf:
`java DeleteLines input.txt output.txt 3-8 7 0 10-15`
Das Programm soll die Datei `input.txt` öffnen, jene Zeilen löschen, welche in Folge beschrieben werden und die Datei wieder unter `output.txt` speichern.
Eine Zeilenbeschreibung kann entweder in der Form `n-m` oder in der Form `n` erfolgen.
Verwende zur Lösung ausschließlich die Klassen `FileInputStream` und `FileOutputStream`.
Damit zeilenweise gelesen werden kann, leite von `FileInputStream` eine Klasse `AsciiInputStream` ab und implementiere in dieser eine Methode `readLine`, welche bis zum nächsten `'\n'` liest.
Achte darauf, dass deine Implementierung auch dann funktioniert, wenn die letzte Zeile ohne ein `'\n'` endet.[1]

2 Pipe

Verwende die Klasse `CopyLowLevel` aus dem Skriptum, um folgende Kommunikationskette aufzubauen. Das Programm kann **nicht** aus IDEs heraus getestet werden.

- Kopiere von `System.in` zeichenweise mit Hilfe von `copySingleByte` in einen `PipedOutputStream`.
- Verbinde den `PipedOutputStream` mit einem `PipedInputStream`.
- Nach Beendigung des Inputs kopiere alle Daten aus dem `PipedInputStream` in 4er-Blöcken mit Hilfe von `copyBuffer` in einen `FileOutputStream`.

Teste das Programm auf der Konsole. Mit `Ctrl+Z` kann EOF simuliert werden.

[1] Punkte, welche ignoriert werden: Ascii kann nur 256 Zeichen darstellen; eine Zeile kann je nach Betriebssystem mit `"\r\n"` oder `"\n"` enden