Modul Verteilte Software WS2022/23
Aufgaben 3

1. Verarbeitung von Byteströmen

Schreiben Sie eine Anwendung mit der die Byteströme einer beliebigen Anzahl von Binärfiles in ein Ausgabefile kopiert werden können. Dabei soll zunächst der Name des Ausgabefiles eingegeben werden. Wiederholt werden nachfolgend die Namen der Eingabefiles angefordert bis ein leerer Dateiname eingegeben wird.
Für den Test des Programms wurden im Übungsordner die mp3-Dateien bereitgestellt. Um das Ergebnis der Zusammenfassung zu überprüfen, spielen Sie die Ausgangdateien und die zusammengefasste Datei mit einem geeigneten Player ab.

Hinweise:

- Zur Eingabe der Dateinamen von Tastatur verwenden Sie die Methode **nextLine** der Klasse **Scanner**. Scanner ist mit dem Objekt System.in zu initialisieren.
- Zur Lesen/Schreiben in die Datei können die Klassen **FileInputStream (FileOutputStream)** und **BufferedInputStream (BufferedOutputStream)** genutzt werden.
- Die Stream-Klassen stellen die Methoden **read(byte[])** und **write(byte[])** zur Verfügung. Beim Erreichen des Dateiendes beträgt die Zahl der gelesenen Zeichen 0. Mit der Methode **close** können die Streams geschlossen werden.

2. Trennen einer Textdatei

Schreiben Sie ein Programm, das eine Textdatei liest und in mehrere Ausgabedateien schreibt. Als Trennzeichen gilt eine definierte Zeichenkette, die auf einer extra Zeile steht. Die Ausgabedateien sollen als Name die fortlaufende Nummer und „.txt“ als Namenserweiterung erhalten.

Definieren Sie dazu die Klasse **TextFileSplitter** mit der Methode **void split()**. Der Dateiname und die Trennzeichenkette sind beim Anlegen eines Objektes der Klasse TextFileSplitter zu übergeben.

Verwenden Sie zum Lesen der Datei die Klassen **BufferedReader** (Methode **readLine** liefert null-String, falls das Ende vom Stream erreicht wurde) und **FileReader** und zum Schreiben die Klasse **FileWriter** (Methode **write**).

Fangen Sie die mögliche FileNotFoundException in einem catch-Block ab.
Testen Sie Ihr Programm mit der Datei „kurzgeschichte.txt“

3. Zerlegen von Eingabeströmen

Kopieren Sie die Textdatei messw.txt in Ihr Projektverzeichnis. Die Datei ist wie folgt aufgebaut

```
Vorbrenner 540 544
Ofen1 800.5  820  809.5
```

usw.

D.h. in der Dateiwerden jeweils nach dem Namen einer Messstelle (z.B. Vorbrenner) eine unterschiedliche Anzahl zur Messstelle gehörender Messwerte erfasst (z.B. 800.5, 820 und 809.5).
Berechnen Sie zu jeder Messstelle den Mittelwert der Messwerte. Die Namen und zugehörigen Mittelwerte sind auf dem Bildschirm aufzulisten.

Verwenden Sie zur Lösung der Aufgabe ein mit dem **FileReader** initialisiertes Objekt der Klasse **Scanner** mit den Methoden
hasNext(), hasNextDouble(), next() und nextDouble().

Zum Einlesen von double-Werten mit Dezimalpunkt muss die „locale"-Einstellung von Scanner geändert werden: useLocale(Locale.US).

4.
Schreiben Sie ein Programm, das mit Hilfe der Methoden der zu erstellenden Klasse (z.B. Analyzer) einen für die Leseanfänger vorgesehenen Text analysiert. Alle im Text vorhandenen Wörter sollen dabei in 2 Kategorien aufgeteilt werden:

1. Wörter, die dem Leseanfänger bereits bekannt sind und
2. alle restlichen Wörter. Die Menge dieser Worte soll erst während der Analyse erstellt werden.

- Erstellen Sie die Menge der bekannten Wörter (ArrayList) als Attribut der Klasse Analyzer und füllen Sie diese mit den Wörtern aus der Datei bekannt.txt in einer dafür vorgesehenen Methode.
- Lesen Sie die Datei „lesetext.txt" wörterweise und speichern Sie die unbekannten Wörter in den ArrayList (ebenfalls Attribut der Klasse Analyzer).