Lösungen Übungsaufgaben Input / Output

1. Dateien und Attribute [PU]

Siehe Musterlösung im Modul FileAttributes

Dies ist ein gutes Beispiel, wie mithilfe der `java.io.File`-Klasse eine hierarchische Baumstruktur (Verzeichnisbaum) traversiert und Attribute der einzelnen Knoten (Dateien/Verzeichnisse) ausgelesen werden können. Problemlos könnte die Anwendung erweitert werden und ein ganzer Teilbaum traversiert und ausgegeben. Dies könnte auch als rekursive Methode ausgeführt werden.

Alternativ könnte das Auslesen des Verzeichnisses und bestimmen der Dateiattribute auch mittels der statischen Methoden in `java.nio.file.Files` erfolgen. Anstelle der `File`-Objekte würde dann mit `Path`-Objekten gearbeitet und die Attribute dieser ermittelt. `Files` erlaubt auch erweiterte Filesystemattribute abzufragen (erweiterte Berechtigungen, Links).
Auch wenn Functional-Streams verwendet werden sollen (siehe PROG2 Functional-Programming), bietet `Files` mehr Möglichkeiten.

Was bedeuteten die Attribute Lesen (`r`), Schreiben (`w`) und Ausführen (`x`) bei einem Verzeichnis?

Ein Verzeichnis kann man sich als spezielle Datei vorstellen, welche eine Liste von Referenzen (`Inodes`) und Metainformationen (Name, Typ, Zugriffsrechte, Grösse, ...) zu den darin enthaltenen Dateien im Filesystem enthält (Inhaltsverzeichnis).
Die Rechte auf Verzeichnisse können entsprechend der Zugriffsrechte auf diese Datei interpretiert werden:

Lesen (`r`)

Das Inhaltsverzeichnis kann gelesen werden, d.h. die Metadaten (Namen & Attribute) der Dateien abgefragt werden.

Schreiben (`w`)

Das Inhaltsverzeichnis kann geschrieben / verändert werden, d.h. es können neue Dateien erstellt oder Dateien gelöscht werden.

Ausführen (`x`)

Es ist möglich das Verzeichnis zu traversieren, d.h. man kann (z.B. mittels `cd`) ins Verzeichnis bzw. durch das Verzeichnis hindurch in Unterverzeichnisse gelangen.

2. Verstehen von Zeichensätzen [PU]

a. Ergänzen Sie in der Klasse `UnderstandingCharsets` den Code, um alle von der JVM unterstützten Zeichensätze auf der Konsole (`System.out`), sowie den für Ihr System definierten Standardzeichensatz auszugeben.
https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/nio/charset/Charset.html

   Siehe Musterlösung im Modul Charsets - Teilaufgabe (a)

b. Ergänzen Sie die Klasse so, dass sie einzelne Zeichen (also Zeichen für Zeichen) im Standardzeichensatz von der Konsole einliest und in zwei Dateien schreibt einmal im Standardzeichensatz und einmal im Zeichensatz `US-ASCII`.
   - Die Eingabe des Zeichens `q` soll das Program ordentlich beenden.
   - Die Dateien sollen `CharSetEvaluation_Default.txt` und `CharSetEvaluation_ASCII.txt` genannt werden und werden entweder erzeugt oder, falls sie bereits existieren, geöffnet und der Inhalt überschrieben.

Siehe Musterlösung im Modul Charsets - Teilaufgabe (b)

Die Musterlösung wurde erweitert, dass sie neben dem Default und ASCII auch explizit Dateien mit den zwei häufigsten Default-Zeichensätze (UTF-8, Windows-1252) generiert.

- Testen Sie Ihr Program mit den folgenden Zeichen: a B c d € f ü _ q
- Öffnen Sie die Textdateien nach Ausführung des Programs mit einem Texteditor und erklären Sie das Ergebnis.
- Öffnen Sie die Dateien anschliessend mit einem HEX-Viewer/Editor und vergleichen Sie.

Im Texteditor sieht man, dass das Zeichen in der Datei mit dem Standardzeichensatz korrekt dargestellt wird, bei der US-ASCII-Datei hingegen wurden die erweiterten Zeichen (€,ü) durch ein Platzhalterzeichen ? ersetzt, da diese in US-ASCII nicht enthalten sind.

Im Hex-Editor ist ersichtlich, dass in der Default-Codierung die Zeichen einen erweiterten Code haben. Entweder ist es ein Wert in der oberen Hälfte der Code-Page (Windows-1252), d.h. grösser 127_{dec} / $7F_{hex}$ (€ → 80_{hex}, ü → FC_{hex}) oder es wird gemäss UTF-8 Codierung zu mehreren Bytes erweitert (€ → E2 82 AC_{hex}, ü → C3 BC_{hex})

3. Byte vs. Zeichen-orientierte Streams [PU]

a. Verzeichnis-Struktur verifizieren: Methode `verifySourceDir()`
   - Das Quell-Verzeichnis soll auf Korrektheit überprüft werden.
   - Korrekt bedeutet, dass das Verzeichnis existiert und ausser zwei Dateien mit den Namen `rmz450.jpg` und `rmz450-spec.txt` nichts weiter enthält.
   - Im Fehlerfall werden Exceptions geworfen.

   Siehe Musterlösung im Modul ByteCharStream - Teilaufgabe (a)

b. Dateien kopieren: Methode `copyFiles()`
   - Jede Datei im Quell-Verzeichnis soll zweimal kopiert werden, einmal zeichen- und einmal byte-orientiert.
   - Dazu soll die jeweilige Datei geöffnet und Element für Element (d.h. byte- bzw. charakterweise) von der Originaldatei gelesen und in die Zieldatei geschrieben werden.
   - Die Kopien sollen so benannt werden, dass aus dem Dateinamen hervorgeht, mit welcher Methode sie erstellt wurde.

   Siehe Musterlösung im Modul ByteCharStream - Teilaufgabe (b)

c. Öffnen Sie die Kopien anschliessend mit einem entsprechenden Programm und erklären Sie die entstandenen Effekte.

   Die Textdokumente können mit jedem Editor geöffnet werden und sind identisch, egal ob sie als byte- oder char-Stream kopiert wurden. Bei der JPG-Datei sieht das anders aus. Die als byte-stream kopierte Datei kann auch weiterhin mit einer Bildanwendung (z.B. Bildvorschau OS, IDE) geöffnet werden. Bei der als character kopierten Datei verweigern die Anwendungen das Bild wegen Kodierfehler zu öffnen.

d. Öffnen Sie die Kopien anschliessend mit einem HEX-Viewer/Editor und erklären Sie die Gründe für die Effekte.

   Der Vergleich der kopierten JPG Dateien im Hex-Editor zeigt, dass die Bytes von Beginn an unterschiedlich sind.

   Der Grund ist, dass einige Bitfolgen keine gültigen Zeichen aus der Code-Page darstellen. Ungültige

Zeichen werden in ein Standard-Zeichen für umgewandelt (bei UTF-8 ist das '  ' resp. 'EF BF BD $_{hex}$'). Damit wird die Datei verändert und entspricht nicht mehr der für JPG gültigen Kodierung.

4. Picture File Datasource [PA]

Die Lösungen zu den bewerteten Pflichtaufgaben erhalten Sie nach der Abgabe und Bewertung aller Klassen.