Praktikum Grundlagen der Programmierung I (Java)

WS 2009/2010

4. Februar 2010

Übung 8.1 Binäres Lesen und Schreiben

In dieser Aufgabe soll erarbeitet werden, wie man Daten binär schreiben und lesen kann.

Betrachten Sie hierzu eine Klasse `Person`, die folgende Attribute besitzt:

- `name` als `String`,
- `alter` als `int`,
- `gewicht` als `double`.

Weiterhin soll die Klasse die entsprechenden Getter und einen passenden Konstruktor besitzen.

Schreiben

Um die Daten als Bytes in einen `OutputStream` zu schreiben, müssen Sie die Methoden `write(byte b)` oder `write(byte[] barray)` verwenden[1].

Verwenden Sie folgendes Binärformat: Als erstes Byte die Länge des Namens in Bytes, dann byteweise die Zeichen des Vornamens, anschließend vier Bytes mit der Binärdarstellung des Alters, acht Bytes mit der Binärdarstellung des Gewichts.

Für die Umwandlung der in `Person` verwendeten Datentypen in Bytes benötigen Sie:

- von der Klasse `String` die Methode `toBytes()`, die einen String als Byte-Array liefert (im vom benutzten Betriebssystem verwendeten Zeichensatz!)
- für den Typ `int` den bitweisen Und-Operator (&) sowie aus der Klasse `Integer` die statische Methode `int rotateRight(int value, int count);`
- für den Typ `double` aus der Klasse `Double` die statische Methode `doubleToLongBits`, die eine 64-Bit Gleitkommazahl binär zu einem `long` macht, sowie aus `Long` die zu oben analoge statische Methode `rotateRight`.

[1] Tatsächlich ist `write(byte b)` als `write(int b)`, akzeptiert also `int`, beachtet aber nur das niedrigste (*low*) Byte.

Tipp: Für das *i*-te Byte eines `int` bzw. `long` können Sie den Ausdruck
`(byte) (Integer.rotateRight(wert,8*i) & 0xFF)` bzw.
`(byte) (Long.rotateRight(wert,8*i) & 0xFF)` verwenden, wobei das 0-te Byte das niedrigste (*low*) Byte ist.

Schreiben Sie so für eine Instanz von `Person` Daten in eine Datei.

Lesen

Versuchen Sie nun, die Daten, die Sie z. B. in eine Datei geschrieben haben, wieder zurückzulesen.

Nun benötigen Sie die Methoden `read(byte[] barray)`, die *n* Bytes in einen Byte-Array der Länge *n* liest, sowie |read()|, die ein Byte liest (dieses wird als `int` zwischen 0 und 255 als Rückgabewert geliefert).

Tipp: Zum Konvertieren der vier Bytes eines Integers können Sie mit der Deklaration `int res=0;` starten, der Anteil des *i*-ten Bytes ergibt sich dann jeweils durch
`res |= Integer.rotateLeft(byte_i & 0xFF,8*i)`

Übung 8.2 DataInputStream und DataOutputStream

Finden Sie heraus, wie Sie mit Hilfe dieser Klassen Ihr Leben einfacher gestalten können.

Betrachten Sie hierzu die Methoden:

- `writeInt(int i)`,
- `writeDouble(double d)`,
- `writeUTF(String s)`, sowie
- `readInt()`,
- `readDouble()`,
- `readUTF()`.

Verwenden des Hex-Editors xvi32

Wenn Sie die Binärdatei betrachten wollen, können Sie den Freeware-Hexeditor xvi32 verwenden (`http://www.chmaas.handshake.de/delphi/freeware/xvi32/xvi32.htm`). Laden Sie hierzu einfach die Zip-Datei herunter und entpacken diese in ein beliebiges Verzeichnis, der Editor ist dann direkt benutzbar (zum Entfernen löschen Sie dieses Verzeichnis dann einfach wieder).

Zum Betrachten der den Bytes entsprechenden Datenwerte können Sie im Menü Tools den Data-Inspector aktivieren. Aktivieren Sie zum Betrachten von 4-Byte-Integer und 8-Byte-Gleitkommazahlen unter Tools, Options im Reiter Data-Inspector die Datentypen longint und IEEE Double. Hier kann auch zwischen Little- und Bigendian gewechselt werden. Welche Endianess verwendet Java beim binären Schreiben mit DataOutputStream?