Aufgaben

Datenströme (Basis: Handout PRG2_OOP6)

1. Gehen Sie nochmals die Kontrollfragen A durch.
   Siehe PRG2_OOP6_KF.docx / PRG2_OOP6_KF.pdf

2. Konsultieren Sie die J2SE API Dokumentation und schreiben Sie von folgenden Klassen je 3 wichtige Methoden auf:
   - FileInputStream
   http://docs.oracle.com/javase/1.4.2/docs/api/java/io/FileInputStream.html
   available()
   read()
   close()

   - DataInputStream
   http://docs.oracle.com/javase/1.4.2/docs/api/java/io/DataInputStream.html
   read()
   readBoolean()
   readByte()
   readChar()
   readDouble()
   readFloat()
   readInt()
   readLong()
   readShort()
   readUnsignedByte()
   readUnsignedShort()
   readUTF()

3. Konsultieren Sie die J2SE API Dokumentation und schreiben Sie von folgenden Klassen je 3 wichtige Methoden auf:
   - BufferedWriter
   http://docs.oracle.com/javase/1.4.2/docs/api/java/io/BufferedWriter.html
   write()
   flush()
   close()

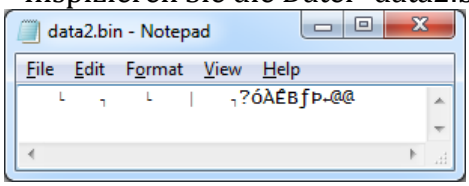
   - PrintWriter
   http://docs.oracle.com/javase/1.4.2/docs/api/java/io/PrintWriter.html
   print()
   println()
   write()
   flush()
   close()

   - FileWriter
   http://docs.oracle.com/javase/1.4.2/docs/api/java/io/FileWriter.html
   keine eigene Methode -, sind alle vererbt, z.B. read(), flush(), close()

Dateien (Basis: Handout PRG2_OOP6, Sourcecode zu den Beispielen)

4. Gehen Sie nochmals die Kontrollfragen B durch.
   Siehe PRG2_OOP6_KF.docx / PRG2_OOP6_KF.pdf

5. Führen Sie die main()-Methode der Klasse WriteBinaryFile2 aus.
   - Inspizieren Sie die Datei "data2.bin" mit einem einfachen Text-Editor/Viewer.

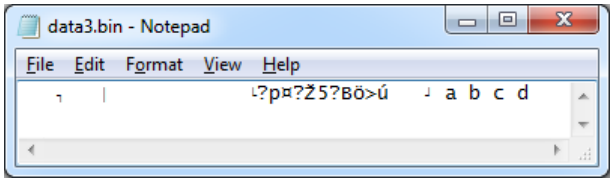


   - Führen Sie die main()-Methode der Klasse ReadBinaryFile2 aus.

```
2
3
5
1.23456789
1000.0
```

6. Schreiben Sie eine neue Klasse WriteBinaryFile3 (basierend auf WriteBinaryFile2), die NEU der Reihe nach 2 int-Werte, 3 float-Werte und 4 char-Werte in eine Datei "data3.bin" hineinschreibt.

```
package oop6_dat6_ss;

import java.io.*;
/**
* @author Felix Rohrer <felix.rohrer@stud.hslu.ch>
 */
public class WriteBinaryFile3 {
    public static void main(String[] args)
    {
        String fileName = "data3.bin";
        File aFile = new File(fileName);
        if ( ! aFile.exists()) {
            try {
                FileOutputStream aFileOutputStream = new FileOutputStream(aFile);
                // DataOutputStream unterstützt elementare Datentypen
                DataOutputStream aDataOutputStream = new DataOutputStream(aFileOutputStream);
                // 1. Anzahl intWerte, 2. intWerte...
                aDataOutputStream.writeInt(2);
                aDataOutputStream.writeInt(5);
                aDataOutputStream.writeInt(9);

                // 1. Anzahl floatWerte, 2. floatWerte ...
                aDataOutputStream.writeInt(3);
                aDataOutputStream.writeFloat(1.23f);
                aDataOutputStream.writeFloat(1.111f);
                aDataOutputStream.writeFloat(123.123f);

                // 1. Anzahl CharWerte, 2. CharWerte ...
                aDataOutputStream.writeInt(4);
                aDataOutputStream.writeChar('a');
                aDataOutputStream.writeChar('b');
                aDataOutputStream.writeChar('c');
                aDataOutputStream.writeChar('d');

                aFileOutputStream.close();
            } catch (IOException e) {
                // ...
            }
        }
    }
}
```



7. Schreiben Sie eine neue Klasse ReadBinaryFile3 (basierend auf ReadBinaryFile2), welche die Datei "data3.bin" bzw. dieses proprietäre File-Format wieder ordentlich einlesen kann.

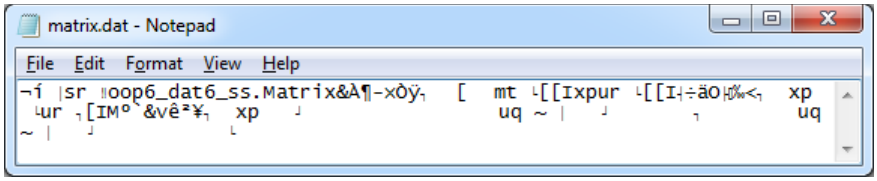
```
package oop6_dat6_ss;
import java.io.*;
/**
 *
 * @author Felix Rohrer <felix.rohrer@stud.hslu.ch>
 */
public class ReadBinaryFile3 {

    public static void main(String[] args)
    {
        String fileName = "data3.bin";
        File aFile = new File(fileName);
        if (aFile.exists()) {
            try {
                FileInputStream aFileInputStream = new FileInputStream(aFile);
                DataInputStream aDataInputStream = new DataInputStream(aFileInputStream);
                // Einlesen der intWerte
                int numberOfInt = aDataInputStream.readInt();
                for (int i = 0; i < numberOfInt; i ++) {
                    System.out.println(aDataInputStream.readInt());
                }
                // Einlesen der floatWerte
                int numberOfFloat = aDataInputStream.readInt();
                for (int i = 0; i < numberOfFloat; i ++) {
                    System.out.println(aDataInputStream.readFloat());
                }
                // Einlesen der charWerte
                int numberOfChar = aDataInputStream.readInt();
                for (int i = 0; i < numberOfChar; i ++) {
                    System.out.println(aDataInputStream.readChar());
                }
                aFileInputStream.close();
            } catch (IOException e) {
                // ...
            }
        }
    }
}

run:
5
9
1.23
1.111
123.123
a
b
c
d
```

Objekt-Serialisierung (Basis: Handout PRG2_OOP6, Sourcecode zu den Beispielen)

8. Gehen Sie nochmals die Kontrollfragen C durch.
   Siehe PRG2_OOP6_KF.docx / PRG2_OOP6_KF.pdf

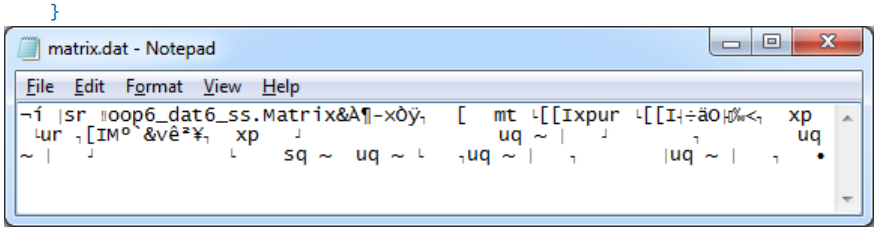
9. Spielen Sie das Beispiel in den Unterlagen selber bzw. führen Sie die main()-Methode der Klasse DemoSerialization aus und inspizieren Sie die Datei "matrix.dat" mit einem Editor.

```
1 0 0 0
0 2 0 0
0 0 3 0
NoOfAcccess: 3
1 0 0 0
0 2 0 0
0 0 3 0
NoOfAcccess: 0
```



10. Erweitern Sie die main()-Methode so, dass NEU 2 Matrizen (unterschiedlich gross) serialisiert und wieder zurück gelesen werden. Hat's geklappt?

```
public static void main(String[] args)
{
    try {
        // Matrix definieren
        Matrix m1 = new Matrix(3, 4);
        m1.setElt(0, 0, 1); // row, col, value
        m1.setElt(1, 1, 2);
        m1.setElt(2, 2, 3);
        m1.print();
        Matrix m2 = new Matrix(2, 2);
        m2.setElt(0, 1, 5);
        m2.setElt(1, 0, 7);
        m2.print();

        // matrix in file schreiben - serialization
        FileOutputStream aFileOutputStream = new FileOutputStream("matrix.dat");
        ObjectOutputStream aObjectOutputStream = new ObjectOutputStream(aFileOutputStream);
        aObjectOutputStream.writeObject(m1);
        aObjectOutputStream.writeObject(m2);
        aFileOutputStream.close();

        // matrix aus file lesen - (de)serialization
        FileInputStream aFileInputStream = new FileInputStream("matrix.dat");
        ObjectInputStream aObjectInputStream = new ObjectInputStream(aFileInputStream);
        Object o = aObjectInputStream.readObject();
        Object o2 = aObjectInputStream.readObject();
        aFileInputStream.close();

        // eingelesens Object ist vom Typ Object -> Cast auf Matrix
        Matrix m3 = (Matrix) o;
        Matrix m4 = (Matrix) o2;
        m3.print();
        m4.print();
    } catch (Exception e) {
        System.out.println("Exception: " + e.getMessage());
    }
}
```



⇨ *Funktioniert*

Binäre und Text Dateien (Basis: Handout PRG2_OOP6, Sourcecode zu den Beispielen)

11. Studieren Sie die Programmier-Pattern im letzten Teil des Handouts.

aFileOutputStream.write(i) ➔ *Byte in den Stream schreiben*

aFileInputStream.read(); ➔ *Byte aus dem Stream lesen*

BufferedWriter aBufferedWriter = new BufferedWriter(aFileWriter);

PrintWriter aPrintWriter = new PrintWriter(aBufferedWriter);

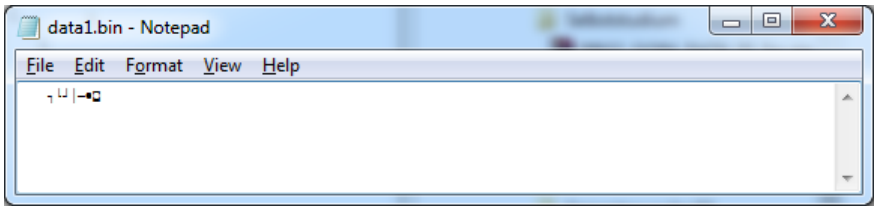
aPrintWriter.println("Dies ist die 1. Zeile.");

➔ *Text via BufferedWriter als Text (ASCII) in Stream schreiben*

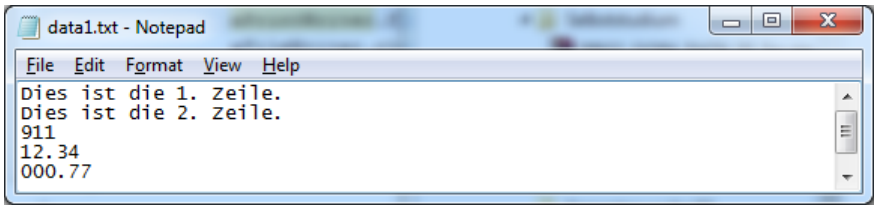
aPrintWriter.flush(); ➔ *Puffer „leeren“ (Daten in File schreiben)*

aFileWriter.close(); ➔ *Stream schliessen (Datei Handle schliessen)*

12. Führen Sie die main()-Methoden aller Klassen aus. Inspizieren Sie zwischendurch die Dateien mit einem Editor.




```
0
1
2
3
4
```




```
Dies ist die 1. Zeile.
Dies ist die 2. Zeile.

911.0
12.34
0.77
```

Ordnung muss sein! (Basis: Handout PRG2_DAT6)

13. Gehen Sie nochmals die Kontrollfragen A, B und C durch.

Siehe PRG2_DAT6_KF.docx / PRG2_DAT6_KF.pdf