Ein-/Ausgabe über Streams

Ein- und Ausgabe in Java verlaufen „stromorientiert“.

Ein Strom (Stream) ist die Verbindung zwischen Programm und Datenquelle/-ziel.

Ein Stream verläuft immer in nur eine Richtung (abstrakte uni-direktionale Verbindung zwischen Programm und „Außenwelt“).

Eingabe:

Das Programm öffnet einen mit einer Datenquelle verbundenen Stream um dann die ankommenden Informationen sequentiell zu lesen.

Ausgabe:

Das Programm öffnet einen Stream zu einem Datenziel um dann Informationen sequentiell in den Stream zu schreiben.

Streams

- Charakter-Streams (Zeichenströme)
- Byte-Streams (Byteströme)

Charakter-Streams:

Datentransport in 16-Bit-Einheiten (mit dem Datentyp char bzw. Unicode-Zeichen). Die Basis-Funktionalität wird durch die abstrakten Klassen Reader und Writer bereitgestellt.

Byte-Streams:

Datentransport in 8-Bit-Einheiten (mit dem Datentype byte). Die Basis-Funktionalität wird durch die abstrakten Klassen InputStream und OutputStream bereitgestellt.

Streams

- Die Basisklassen befinden sich im Paket java.io
- Alle abgeleiteten Klassen der vier Basisklassen enthalten den Namen der Basisklasse im Namen.
- Verkettung von Streams um z.B. Übergänge von Bytes-Streams in Character-Streams zu ermöglichen (FilterStreams)

Abstrakte Klasse Reader

- public abstract int read()

  Liefert das nächste Zeichen aus dem Reader-Objekt als int-Wert

- public int read (char[] c)

  Füllt c mit Zeichen aus dem Reader-Objekt und liefert die Anzahl der gelesenen Zeichen

- public int read (char[] c, int off, int n)

  Füllt c mit Zeichen aus dem Reader-Objekt ab dem Index off und liefert die Anzahl der gelesenen Zeichen

- public void close()

  schließt den Strom

Abstrakte Klasse Writer

- public abstract void write (int c)

  schreibt das Zeichen c in das Writer-Objekt
- public void write (char[] c)

  schreibt die in c gespeicherten Zeichen in das Writer-Objekt
- public int write (char[] c, int off, int n)

  schreibt die in c gespeicherten n Bytes ab Index off in das Writer-Objekt
- public void write (String s)

  schreibt die in s gespeicherten Zeichen in das Writer-Objekt
- public void write (String s, int off, int n)
- public void close()

  schließt den Strom
- public void flush()

  leert einen evt. Puffer durch sofortige Abarbeitung aller noch anstehenden Zeichen

Klasse System

- Klasse System stellt für drei Byte-Streams die Klassenvariablen in, out und err bereit

- Der Eingabestrom System.in ist standardmäßig mit der Tastatur verbunden, während die Ausgabeströme System.out und System.err in das Konsolenfenster schreiben

Lesen aus CharacterStreams

```java
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

public class Lesen {
    public static void main(String[] args) {
        BufferedReader br;
        try {
            br = new BufferedReader(new FileReader("Test1.txt"));
            String s;
            while ((s = br.readLine()) != null) {
                System.out.println(s);
            }
            br.close();
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

Schreiben in CharacterStreams

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

public class Schreiben {

    public static void main(String[] args) {
        FileWriter fw;
        try {
            fw = new FileWriter("Test1.txt", true);
            BufferedWriter bw = new BufferedWriter(fw);
            bw.write("Samstag\n");
            bw.close();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```