Ein- und Ausgabe mit den IO-Tools

Die nativen (eingebauten) Ein- und Ausgabefunktionen der Java-API sind zwar recht umfangreich und für viele Zwecke anpassbar - ihre Verwendung in einem Java-Programm verlangt aber beträchtliche Einarbeitung und ist für Einsteiger eher verwirrend. Der Code einfacher Übungsprogramme kann leicht unübersichtlich werden.

Für Übungsprogramme sind eine Reihe vorgefertigter Klassen verfügbar, die alle grundlegenden Input / Output-Funktionalitäten in leicht verwendbarer Form anbieten. Für Konsolenprogramme können die von Herrn Prof. Mössenböck (Universität Graz / Österreich) freundlicherweise zur Verfügung gestellten IO-Tools (aus dem empfehlenswerten Lehrbuch *Sprechen Sie Java?*) verwendet werden.

http://www.ssw.uni-linz.ac.at/Misc/JavaBuch/#InOut

Eine leicht angepaßte Version ist über meine Webseite erhältlich:

http://skripta.de/java1/IO/IO.zip

Beschreibung

Es gibt im Computer drei Hardware-Bestandteile, für die IO-Funktionen vorhanden sind:

- **Tastatur** (Eingabe)
- **Bildschirm** (Ausgabe)
- **Datei** (Ein- und Ausgabe)

(Mausereignisse spielen erst in der GUI-Programmierung eine Rolle)

Für jeden Datentyp (siehe unten) gibt es eigene In- und Out-Methoden. Es handelt sich um statische Methoden, die über den Klassennamen angesprochen werden (**In**.read.. bzw. **Out**.print…).

Klasse IN (Einlesen von Tastatur bzw. von Datei):

```
boolean b = In.readBoolean();   // liest true oder false
char    c = In.readChar();      // liest einen Buchstaben
double  d = In.readDouble();    // liest eine double-Zahl
float   f = In.readFloat();     // liest eine float-Zahl
int     i = In.readInt();       // liest eine int-Zahl
long    l = In.readLong();      // liest eine long-Zahl
String  s = In.readLine();      // liest einen String (inkl.Leerzeichen)
String  s = In.readString();    // liest einen String (mit „")
String  s = In.readWord();      // liest ein einzelnes Wort
boolean b = In.done();          // gibt Erfolg oder Misserfolg
void    In.fflush();            // leert Tastaturpuffer
char    c = In.peek();          // prüft folgendes Zeichen
void    In.stop()               // stoppt Bildschirmausgabe

void    open(String);           // öffnet Datei
void    close();                // schliesst Datei
String  s = In.readFile();      // liest ganze Datei in einen String
```

Verwendungsbeispiel:

```
. . .
System.out.println("Geben Sie einen String ein: ");
String s = In.readLine();
System.out.println("Ihre Eingabe war: " + s);
. . .
```

Hinweis: `In.readLine()` ist zum Einlesen von Strings besser geeignet als `In.readString()` oder `In.readWord()`

Dateifunktionen:

Nach der Anweisung **`open(„Dateiname")`** wird die Ein- und Ausgabe auf die angegebene Datei umgelenkt. Alle folgenden Befehle beziehen sich dann auf diese Datei. Die Anweisung **`close()`** schliesst die Datei und alle folgenden Befehle beziehen sich wieder auf die Tastatur (siehe Beispiel unten).

Klasse **OUT** (Ausgabe auf Bildschirm bzw. in Datei):

Ohne Zeilenvorschub:

```
void print(boolean);
void print(char);
void print(int);
void print(long);
void print(float);
void print(double);
void print(String);
void print(Object);
```

Mit Zeilenvorschub:

```
void println(boolean);
void println(char);
void println(int);
void println(long);
void println(float);
void println(double);
void println(String);
void println(Object);
```

Ausgabe-Umlenkung :

```
void open(String);
void close();
```

In Java gibt es keinen Befehl zum blockweisen Schreiben wie in C / C++. Die Ausgabe orientiert sich an Datentypen und hängt die Daten aneinander bis zur close()-Anweisung. Die **`done()`**-Methode überprüft das Ergebnis der vorangegangene Aktion (Eingabe oder Ausgabe) und gibt true oder false zurück.

Beispiele für die Verwendung der IO-Tools

BEISPIEL 1

```
//////////////////////////
// File-Funktionen:
// Einlesen einer Zeile von Tastatur
// Speichern der Zeile in Datei
// Lesen aus Textdatei und anzeigen
// auf Bildschirm

class FileTest01 {

   public static void main(String [] args) {

     Out.println("FileTest: Geben Sie einen Text ein: ");

     String s = In.readLine(); // Zeile von Tastatur einlesen

     Out.open("c:test.txt");          // Datei oeffnen
     Out.println(s);                  // Text in Datei schreiben
     Out.close();                     // Datei schliessen

     if(Out.done()==true)             // Ergebnis pruefen
       Out.println("Datei geoeffnet und Text geschrieben");
     else
       Out.println("Datei konnte nicht geoeffnet werden");

     In.open("c:test.txt");           // Datei zum Lesen oeffnen
     String t = In.readFile(); // Gesamte Datei in String lesen
     In.close();                      // Datei schliessen

     if(In.done()==true)              // Ergebnis pruefen
       Out.println("Datei geoeffnet und Text gelesen");
     else
       Out.println("Datei konnte nicht geoeffnet werden");

     Out.println(t);                  // String anzeigen
     In.stop();
   }
}
```

BEISPIEL 2

```
//////////////////////////
// Objekte in einem Vektor erzeugen
// In Datei schreiben - aus Datei lesen

class Student01 {

     int    matrikelnr;   // Attribute
     double einkommen;
     String vorname, nachname;

    // KONSTRUKTOR:

     public Student01() {

        this.matrikelnr = 1;
        this.einkommen  = 2000;
        this.vorname    = "default";
```

```
        this.nachname   = "default";
        Out.println("Konstruktor aufgerufen...");
    }

    public void gibAus() {  // Objekt-Attribut ausgeben

        Out.println("\nStudent:  " + vorname + " " + nachname);
        Out.println("Einkommen:  " + einkommen);
        Out.println("MatrikelNr: " + matrikelnr);
    }

    ////////////////////////////////////////////////////////////

    public static void main(String args[]) {

     Out.print("Wieviele Studenten ?");
     int anzahl = In.readInt();

     Student01[] vektor = new Student01 [anzahl]; // Vektor

     for(int i=0; i < vektor.length; i++)
            vektor[i] = new Student01();  // Objekte erzeugen

     for(int i=0; i < vektor.length; i++)
            vektor[i].gibAus();           // anzeigen


    ////////////////////////////////////////////////

    Out.open("c:test.dat");         // Datei oeffnen

    for(int i=0; i < vektor.length; i++) {  // in Datei schreiben

            Out.println(vektor[i].vorname + " " + vektor[i].nachname);
            Out.println(vektor[i].matrikelnr);
            Out.println(vektor[i].einkommen);
    }
    Out.close();

    if(Out.done()==true)            // Ergebnis pruefen
      Out.println("Datei geoeffnet …");
    else
      Out.println("Datei konnte nicht geoeffnet werden");

    ////////////////////////////////////////////////

    In.open("c:test.dat");          // Datei oeffnen
    String s=In.readFile();         // Ganze Datei lesen
    In.close();                     // Datei schliessen

    if(Out.done()==true) {          // Ergebnis pruefen
      Out.println("Datei geoeffnet und Text gelesen");

      Out.println(s);               // Daten als String ausgeben

    } else
      Out.println("Datei konnte nicht geoeffnet werden");

    In.stop();
  }
}
```

BEISPIEL 3 (mit static-Methoden)

```
//////////////////////////
// Objekte in einem
// Vektor erzeugen,
// In Datei schreiben
// Aus Datei lesen

class Student01 {

     int    matrikelnr;   // Objekt-Attribute
     double einkommen;
     String vorname, nachname;

     static int anzahl;     // Klassen-Attribute
     static int matrik;
     static Student01[] vektor;

    // KONSTRUKTOR ist die einzige Objektmethode ! //////////

     public Student01() {
        matrik++;               // Zaehler
        this.matrikelnr = matrik;
        this.einkommen  = 2000;
        this.vorname    = "default";
        this.nachname   = "default";
        Out.println("Konstruktor aufgerufen...");
     }

     ////////////////////////////////////////////////
     // Alles andere sind Klassenmethoden

     public static void makeVector() {

        Out.print("Wieviele Studenten ?");
        anzahl = In.readInt();
        In.read();

        vektor = new Student01 [anzahl]; // Vektor erzeugt

        for(int i=0; i < vektor.length; i++) {
            vektor[i] = new Student01();  // Objekte erzeugt
        }
        Out.println("Vektor mit " + anzahl + " Objekten erzeugt\n");
     }

    ////////////////////////////////////////////////

     public static void showVector() {  // Objekt-Attribute ausgeben

        Out.println("----------------------------");
        Out.println(vektor.length + " Objekte im Vektor:");

        for(int i=0; i < vektor.length; i++) {

        Out.println("\nStudent: " + vektor[i].vorname + " "
                                  + vektor[i].nachname);
        Out.println("Einkommen: " + vektor[i].einkommen);
        Out.println("MatrikelNr:" + vektor[i].matrikelnr);
        }
     }
```

```
        ////////////////////////////////////////////////

        public static void writeFile() {

           Out.println("----------------------------");
           Out.println("In Datei geschrieben...\n");

           Out.open("c:test.dat");

           for(int i=0; i < vektor.length; i++) {

               Out.println(vektor[i].vorname + " " + vektor[i].nachname);
               Out.println(vektor[i].matrikelnr);
               Out.println(vektor[i].einkommen);
           }
           Out.close();
        }
        ////////////////////////////////////////////////

        public static void readFile() {

           In.open("c:test.dat");
           String s=In.readFile();
           In.close();

           Out.println("--------------------------");
           Out.println("Aus Datei eingelesen");
           Out.println(s);
           In.read();
        }
        ////////////////////////////////////////////////////////7

        public static void main(String args[]) {


         makeVector();

         showVector();

         writeFile();

         readFile();

         In.read();
      }
   }
```