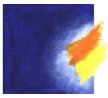


Java – Dateiverarbeitung

Bislang werden Ein- und Ausgaben, Ergebnisse usw. nur so lange gespeichert, wie unser Programm abläuft. Oft ist es aber notwendig, oder zumindest gewünscht, Daten längerfristig zu speichern. Grundsätzlich bieten sich hier eine Datenbank oder eine Datei an.

Die Ein- und Ausgabe in eine Datei wird im Folgenden erläutert. Zu beachten ist, dass es sich hier nicht um die Verarbeitung von „byte-verarbeiteten" Dateien handelt, also z.B. Word-Dateien, MP3s oder PDF-Dokumente. Hierfür gibt es ggf. weitere Klassen. Die folgende Möglichkeit bezieht sich auf reine Binär- oder Zeichenbasierender Verarbeitung, also z.B. Text-Dateien, aber auch HTML-Dokumente.

Die notwendigen Methoden sind im Java-IO-Paket enthalten, welches durch
`import java.io.*`
eingebunden wird.

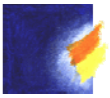
Dateien werden in sogenannten „Streams" verarbeitet, durch das Einbinden des IO-Pakets stehen die neuen Datentypen `FileInputStream` (Einlesen einer Datei) und `FileOutputStream` (Schreiben einer Datei) zur Verfügung:

```
FileInputStream lies = new FileInputStream("speicher.txt");
FileOutputStream schreib = new FileOutputStream("speicher.txt")
```

Zum Einlesen steht nun z.B. der Befehl `lies.read()` und zum Speichern der Befehl `schreib.write()` zur Verfügung.

Ein einfaches Beispiel könnte zunächst so aussehen:

```
import java.io.*;
import java.util.Scanner;

public class DateiSchreibenLesen
{
  public static void main(String args[]) throws IOException
  {
    FileOutputStream schreib = new FileOutputStream("speicher.txt");
    FileInputStream lies = new FileInputStream("speicher.txt");
    Scanner eingabe = new Scanner(System.in);

    String zeile;
    int z=0;

    zeile = eingabe.next();
    schreib.write(zeile.getBytes(),0,zeile.length());
    schreib.close();

    while((z=lies.read())!=-1){

      System.out.print((char)z);

    }
    lies.close();
  }
}
```



Das Problem ist nun, dass Fehlermeldungen erscheinen:

```
Compiliere H:\Java\DateiSchreibenLesen.java mit Java-Compiler
DateiSchreibenLesen.java:8:32: error: unreported exception FileNotFoundException; must be caught or declared to be thrown
    FileOutputStream schreib = new FileOutputStream("line.txt");
                               ^
[…]
    while((z=lies.read())!=-1){
                  ^
DateiSchreibenLesen.java:24:17: error: unreported exception IOException; must be caught or declared to be thrown
      lies.close();                      //Datei wieder schließen
                ^
6 errors
```

Die Fehlermeldung sagt aus, dass Java bei der Verarbeitung von Datei-Streams die damit verbundenen möglichen Fehlermeldungen (`FileNotFoundException` und `IOException`) nicht einfach ausgeben kann, sondern sie müssen „gefangen“ (caught) oder „zum (weg)werfen bestimmt“ (declared to be thrown) werden.

Aufgabe 1

Kopiere das oben gezeigte Beispiel zum Erstellen und Lesen einer Text-Datei in eine Java-Datei und ergänze die notwendigen Reaktionen auf die Fehler `FileNotFoundException` und `IOException`.

a) Hänge hierzu zunächst einen `throws`-Befehl an die Klasse `main` an.

Jetzt funktioniert zwar das Programm, jedoch erscheint immer noch die entsprechende Exception. Daher:

b) Ergänze deinen Programmcode um `try` und `catch` für beide Exceptions und lass eine entsprechend passende Meldung ausgeben.

Aufgabe 2

Analysiere den Aufbau des Programms, welche Befehle führen zum Schreiben, bzw. Lesen der Datei.

Aufgabe 3

a) Es sollen nacheinander zehn Eingaben abgefragt und gemeinsam in der Datei „speicher.txt“ gespeichert werden. Verändere deinen Code so, dass diese Abfragen inkl. Speichern durchgeführt werden. Schau nach den zehn Eingaben mal in deine Datei speicher.txt, wie sie dort gespeichert wurden.

b) Ergänze an deine Eingabe die Zeichenfolge „`\n`“ (New Line) und führe das Programm nochmal aus. Wie hat sich der Inhalt der Datei und auch die Ausgabe verändert?
Öffne die Datei auch mal per rechter Maustaste „Öffnen mit“ → „Wordpad“