Typische Konstrukte für das Schreiben und Lesen von Dateien

1. Dateien für einfache Typen wie einzelne Buchstaben, Zahlen o.ä. – hier für ganze Zahlen:

Schreiben

```
try                                  //
{
  //Öffnen einer Datei zum (Über-)Schreiben
  BuffreredWriter aufDisk = new BufferedWriter (
                                  new FileWriter("xyz.abc"));

  Wiederholstruktur, um immer wieder zu schreiben:
    aufDisk.write (zahl); //Schreiben einer Komponente


  aufDisk.close(); //Schließen der Datei
}
catch (IOException ex)
{
   //Reaktion auf evtl. Fehler (oder auch gar nichts)
}
```

Lesen

```
try                                  //
{
  //Öffnen einer (vorhandenen) Datei zum Lesen
  BufferedReader vonDisk = new BufferedReader (
                                  new FileWriter("xyz.abc"));

  do {
    zahl = vonDisk.read();  //Lesen einer Komponente
  } while (zahl != -1); //-1 zeigt Ende der Datei an

  vonDisk.close(); //Schließen der Datei
}
catch (IOException ex)
{
   //Reaktion auf evtl. Fehler (oder auch gar nichts)
}
```

2. Arbeit mit Textdateien, wobei typischerweise der Text als ein String mit eingeschlossenen Zeilenumbrüchen (\n) geschrieben und zeilenweise gelesen wird: try/catch und Schließen wie oben.

```
BufferedWriter aufDisk  = new BufferedWriter(
    new FileWriter (“Datei.txt“));  //Definition & Öffnen

aufDisk.write (ganzerText);  //alles auf einmal...
  //...oder mehrfach: aufDisk.write (eineZeile);
  //Extra-Zeilenumbruch auch mit aufDisk.newLine();
```

```
LineNumberReader vonDisk = new LineNumberReader(
   new BufferedReader(new FileReader("Datei.txt")));

zeile = vonDisk.readLine(); //Lesen einer Zeile (String)
nummer = vonDisk.getLineNumber(); //Abfragen der
    Zeilennummer (int) der gerade gelesenen Zeile
Ist die Datei zu Ende, wird  zeile = null
```

3. Dateien, die beliebige (eigene) Objekte aufnehmen können, hier z.B. Autos (wobei die Kfz-Klasse dazu allerdings mit dem Zusatz *public class Kfz* ***implements java.io.Serializable*** erklärt werden muss):

```
try
{
  oos = new ObjectOutputStream(
    new FileOutputStream ("name.dat"));


  //folgenden Befehl wiederholen, um mehrere Autos zu
  //schreiben (auto sei ein Objekt vom Typ Kfz)
  oos.writeObject (auto); //schreibt ein Objekt
  //(auch writeInt, writeDouble oder writeBoolean mögl.)

  oos.close();
}
catch (IOException ex)
{..}
```

```
try  //Fehler, falls Datei nicht vorhanden oder close ver-
sagt
{
  ois = new ObjectInputStream(
    new FileInputStream ("name.dat"));
  try //Fehler, wenn Datei zu Ende
  {
    //folgenden Lesebefehl immer wieder wiederholen:
    auto = (Kfz) ois.readObject();  //liest nächstes Auto
  }                                          //oder readInt,..
  catch (EOFException eex) //EOF = End of File =..
  {                                          //..= Dateiende
     ois.close();  //Datei schließen, da zu Ende
  }
}
catch (Exception ex) //für IO- oder ClassNotFound-Ex.
{..}
```

Die Art der Wiederholstrukturen richtet sich danach, wie die einzelnen Komponenten zur Verfügung stehen. Passende Variable sollten vorher definiert (und vorm Schreiben mit Inhalt gefüllt) sein, während sie beim Lesen gefüllt werden. Achtung: Wird beim Dateiende -1 oder null gemeldet, sollte diese Komponente nicht mehr verwendet werden, weil schon über das Dateiende hinaus gelesen wurde.