Teil IX

Datenspeicherung und -transfer – Java Streams

Streams: Interfaces

Try with resources

- `java.io.AutoCloseable`: void close() throws Exception
  - ∟ `java.io.Closeable`
- `java.io.Flushable`: void flush() throws IOException
- `java.io.Appendable`
- `java.io.Readable`

Streams: Lesen

Zeichen-basiertes Lesen

```
- java.io.Reader
  ∟ java.io.BufferedReader
    ∟ java.io.LineNumberReader
  ∟ java.io.InputStreamReader
    ∟ java.io.FileReader
```

implements:

- `Closeable`
- `AutoCloseable`
- `Readable`

Streams: Lesen

Zeichen-basiertes Lesen

```
- java.io.Reader
  ∟ java.io.BufferedReader
    ∟ java.io.LineNumberReader
  ∟ java.io.InputStreamReader
    ∟ java.io.FileReader
```

implements:

- `Closeable`
- `AutoCloseable`
- `Readable`

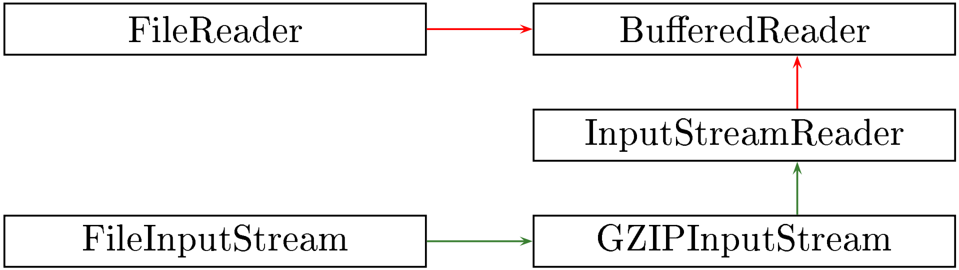
Byte-basiertes Lesen

```
- java.io.InputStream
  ∟ java.io.FilterInputStream
    ∟ java.io.BufferedInputStream
    ∟ java.io.LineNumberInputStream
    ∟ java.util.zip.InflaterInputStream
      ∟ java.util.zip.GZIPInputStream
      ∟ java.util.zip.ZipInputStream
  ∟ java.io.FileInputStream
```

implements:

- `Closeable`
- `AutoCloseable`

Streams: Lesen



Streams: Daten lesen

```java
public static List<String> readLines(
  String fileName
) {
  List<String> result = new ArrayList<>();
  File file = new File(fileName);
  try (
    FileReader fileReader
      = new FileReader(file);






    BufferedReader bufferedReader
      = new BufferedReader(fileReader);
  ) {
    readLine(bufferedReader, result);
  } catch (FileNotFoundException fnfEx) {
    fnfEx.printStackTrace(System.err);
  } catch (IOException ioEx) {
    ioEx.printStackTrace(System.err);
  } finally {
    return result;
  }
}
```

Streams: Daten lesen

```java
public static List<String> readLines(
  String fileName
) {
  List<String> result = new ArrayList<>();
  File file = new File(fileName);
  try (
    FileReader fileReader
      = new FileReader(file);






    BufferedReader bufferedReader
      = new BufferedReader(fileReader);
  ) {
    readLine(bufferedReader, result);
  } catch (FileNotFoundException fnfEx) {
    fnfEx.printStackTrace(System.err);
  } catch (IOException ioEx) {
    ioEx.printStackTrace(System.err);
  } finally {
    return result;
  }
}
```

```java
public static List<String>
     readLinesCompressed(
  String fileName
) {
  List<String> result = new ArrayList<>();
  File file = new File(fileName);
  try (
    FileInputStream fis
      = new FileInputStream(file);
    BufferedInputStream bis
      = new BufferedInputStream(fis);
    GZIPInputStream gzis
      = new GZIPInputStream(bis);
    InputStreamReader isr
      = new InputStreamReader(gzis);
    BufferedReader bufferedReader
      = new BufferedReader(isr)
  ) {
    readLine(bufferedReader, result);
  } catch (FileNotFoundException fnfEx) {
    fnfEx.printStackTrace(System.err);
  } catch (IOException ioEx) {
    ioEx.printStackTrace(System.err);
  } finally {
    return result;
  }
}
```

Streams: Daten lesen

```java
private static void readLine(
  BufferedReader bufferedReader,
  List<String> result
) throws IOException {
  String line = bufferedReader.readLine();
  while (line != null) {
    result.add(line);
    line = bufferedReader.readLine();
  }
}
```

Streams: Zeichen-basiertes Schreiben

java.io.Writer

- implements: `Closeable`, `Flushable`, `Appendable`, `AutoCloseable`
- `abstract void close() throws IOException`
- `abstract void flush() throws IOException`
- `void write( <Typ> arg ) throws IOException`

java.io.PrintWriter `extends Writer`

- `void close()`
- `void flush()`
- `void write( <Typ> arg )`
- `void print( <Typ> arg )`
- `void println( <Typ> arg )`

$\rightarrow$ keine Exceptions

Streams: Zeichen-basiertes Schreiben

java.io.OutputStreamWriter `extends Writer`

java.io.FileWriter `extends OutputStreamWriter`

java.io.BufferedWriter `extends Writer`

Streams: Byte-basiertes Schreiben

java.io.OutputStream

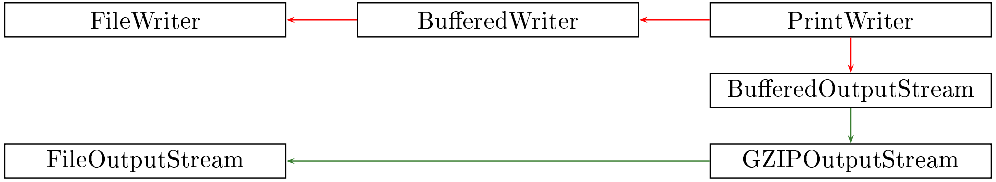
- implements: `Closeable`, `Flushable`, `AutoCloseable`
- `void close() throws IOException`
- `void flush() throws IOException`
- `abstract void write()`

java.io.FilterOutputStream `extends OutputStream`

java.io.BufferedOutputStream `extends FilterOutputStream`

java.util.zip.DeflaterOutputStream `extends FilterOutputStream`
java.util.zip.GZIPOutputStream `extends DeflaterOutputStream`
java.util.zip.ZipOutputStream `extends DeflaterOutputStream`

Streams: Schreiben



Streams: Daten schreiben

```java
public static void writeLines(
    String fileName,
    List<String> lines
) {
  File file = new File(fileName);
  try (
    FileWriter fw = new FileWriter(file);



    BufferedWriter bw = new BufferedWriter
     (fw);

    PrintWriter pw = new PrintWriter(bw);
  ) {
    writeLine(pw, lines);
  } catch (FileNotFoundException fnfEx) {
    fnfEx.printStackTrace(System.err);
  } catch (IOException ioEx) {
    ioEx.printStackTrace(System.err);
  }
}
```

Streams: Daten schreiben

```java
public static void writeLines(
    String fileName,
    List<String> lines
) {
  File file = new File(fileName);
  try (
    FileWriter fw = new FileWriter(file);



    BufferedWriter bw = new BufferedWriter
     (fw);

    PrintWriter pw = new PrintWriter(bw);
  ) {
    writeLine(pw, lines);
  } catch (FileNotFoundException fnfEx) {
    fnfEx.printStackTrace(System.err);
  } catch (IOException ioEx) {
    ioEx.printStackTrace(System.err);
  }
}
```

```java
public static void writeLinesCompressed(
  String fileName,
  List<String> lines
) {
  File file = new File(fileName);
  try (
    FileOutputStream fos
      = new FileOutputStream(file);
    GZIPOutputStream gzos
      = new GZIPOutputStream(fos);
    BufferedOutputStream bos
      = new BufferedOutputStream(gzos);
    PrintWriter pw = new PrintWriter(bos);
  ) {
    writeLine(pw, lines);
  } catch (FileNotFoundException fnfEx) {
    fnfEx.printStackTrace(System.err);
  } catch (IOException ioEx) {
    ioEx.printStackTrace(System.err);
  }
}
```

Streams: Daten schreiben

```java
private static void writeLine(
  PrintWriter pw,
  List<String> lines
) {
  for (String line : lines) {
    pw.println(line);
  }
}
```