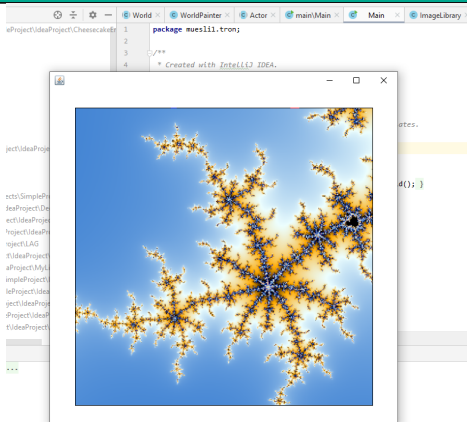


FOP Recap #13



TECHNISCHE
UNIVERSITÄT
DARMSTADT

File-IO





Moin Moin!



File-IO

Wie? Was? IO?

Habe ich da Stream gehört?

`BufferedReader readLine`

`BufferedWriter write`

mit Streams

IOException?

String

String <-> Zahl



- IO steht für Input/Output
- Heißt: Lesen und Schreiben von Dateien
- Unterschiedliche Typen:
 - ▣ Textdateien (z.B. .txt)
 - ▣ Bilddateien (z.B. .png, .jpg)
 - ▣ Sounddateien (z.B. .ogg, .mp3)
 - ▣ Videodateien (z.B. .mp4)
 - ▣



■ Typische Klassen mit denen man hier arbeitet:

- ▣ File
- ▣ InputStream
- ▣ OutputStream
- ▣ Reader
- ▣ Writer
- ▣ IOException
- ▣

File-IO

Habe ich da Stream gehört?



TECHNISCHE
UNIVERSITÄT
DARMSTADT

- InputStream und OutputStream waren zuerst da!
- Teilen sich dieselbe Grundidee: Funktionieren wie ein Fließband
- Haben jedoch nichts miteinander zu tun!

File-IO

BufferedReader readLine – Gut für Textdateien



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
1 // ! Absoluter Pfad ist nicht empfehlenswert !
2 File myFile = new
  ↪ File("C:\\Users\\Marc\\Daten\\TestVerzeichnis\\TextDatei.txt");
3 FileReader fileReader = new FileReader(myFile);
```

```
1 // Relativer Pfad zum Programm
2 FileReader fileReader = new FileReader("TextDatei.txt");
3
4 BufferedReader bufferedReader = new BufferedReader(fileReader);
5
6 String firstLine = bufferedReader.readLine();
7
8 bufferedReader.close();
```

File-IO

BufferedWriter write — Gut für Textdateien



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
1 // Relativer Pfad zum Programm
2 FileWriter fileWriter = new FileWriter("TextDatei.txt");
3
4 BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);
5
6 bufferedWriter.write("First line of the document!\n");
7
8 bufferedWriter.close();
```


File-IO

BufferedWriter write — Gut für Textdateien



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
1 // Relativer Pfad zum Programm
2 FileWriter fileWriter = new FileWriter("TextDatei.txt");
3 BufferedWriter writer = new BufferedWriter(fileWriter);
4
5 writer.write(String.valueOf(true));
6 writer.write(",")
7 writer.write(String.valueOf(5.2));
8 writer.write(",")
9 writer.write(String.valueOf(10));
10
11 writer.close();
```



```
1 Stream<String> stream = Files.lines(Paths.get("TextDatei.txt"));
2
3 List<String> lines = stream.collect(Collectors.toList());
4
5 stream.close();
```

Das steht heute auf dem Plan



TECHNISCHE
UNIVERSITÄT
DARMSTADT

File-IO

IOException?

Das Problem

try-with-resources

String

String <-> Zahl

IOException?



```
1 try {
2     FileReader fileReader = new FileReader("TextDatei.txt");
3
4     BufferedReader bufferedReader = new
        ↳ BufferedReader(fileReader);
5
6     String firstLine = bufferedReader.readLine();
7
8     // !!!
9     bufferedReader.close();
10 }
11 catch(IOException e) {
12     e.printStackTrace();
13 }
```

IOException?

Das Problem



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
1  FileReader reader = null;
2  try {
3      reader = new FileReader("TextDatei.txt");
4      // ....
5  }
6  catch(IOException e) {
7      e.printStackTrace();
8  }
9  finally {
10     // Close Reader
11 }
```

IOException?

Das Problem



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
1  FileReader reader = null;
2  ....
3  finally {
4      if(reader != null) {
5          try {
6              reader.close();
7          }
8          catch(IOException e) {
9              e.printStackTrace();
10         }
11     }
12 }
```

IOException?

try-with-resources



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
1 try(FileReader fileReader = new FileReader("TextDatei.txt");
2     BufferedReader bufferedReader = new
    ↪     BufferedReader(fileReader)) {
3
4     String firstLine = bufferedReader.readLine();
5
6 }
7 catch(IOException e) {
8     e.printStackTrace();
9 }
```

IOException?

try-with-resources



TECHNISCHE
UNIVERSITÄT
DARMSTADT

```
1 try(BufferedReader bufferedReader = createANewReader()) {  
2  
3     String firstLine = bufferedReader.readLine();  
4  
5 }  
6 catch(IOException e) {  
7     e.printStackTrace();  
8 }
```


Das steht heute auf dem Plan



TECHNISCHE
UNIVERSITÄT
DARMSTADT

File-IO

IOException?

String
 split
 contains
 indexOf
 substring

String <-> Zahl



```
1 String text = "Dieser Text ist toll!";  
2  
3 String[] split = text.split(" ");  
4  
5 System.out.println(Arrays.toString(split));  
6 // -> [Dieser, Text, ist, toll!]
```



```
1 String text = "A,B,C,";  
2  
3 String[] split = text.split(",");  
4  
5 System.out.println(Arrays.toString(split));  
6 // -> [A, B, C]
```



```
1 String text = "A,B,C,";  
2  
3 System.out.println(text.contains("B"));  
4 // -> true  
5 System.out.println(text.contains("Z"));  
6 // -> false
```



```
1 String text = "A,B,C,";  
2  
3 System.out.println(text.indexOf("B"));  
4 // -> 2  
5 System.out.println(text.indexOf("Z"));  
6 // -> -1  
7 System.out.println(text.indexOf(", "));  
8 // -> 1
```



```
1 String text = "0123456789";  
2  
3 String a = text.substring(1);  
4 // -> 123456789  
5 String b = text.substring(2);  
6 // -> 23456789  
7 String c = text.substring(2, 5);  
8 // -> 234
```

Das steht heute auf dem Plan



TECHNISCHE
UNIVERSITÄT
DARMSTADT

File-IO

IOException?

String

String <-> Zahl



```
1 String text = "257";  
2  
3 // NumberFormatException?  
4 int num = Integer.parseInt(text);  
5  
6 String other = "25.112";  
7  
8 // NumberFormatException?  
9 double otherNum = Double.parseDouble(other);
```




```
1  int num = 257;  
2  
3  String text = String.valueOf(num);  
4  
5  double otherNum = 25.112;  
6  
7  String other = String.valueOf(otherNum);
```



Live-Coding!