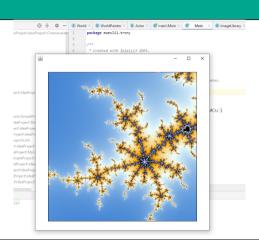
### FOP Recap #13



#### File-IO



### Moin Moin!

#### Das steht heute auf dem Plan



File-IO
Wie? Was? IO?
Habe ich da Stream gehört?
BufferedReader readLine
BufferedWriter write

IOException?

mit Streams

String

## File-IO Wie? Was? IO?



- 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)
  - Videodatein (z.B. .mp4)
  - ....

Wie? Was? IO?



- Typische Klassen mit denen man hier arbeitet:
  - □ File
  - □ InputStream
  - OutputStream
  - Reader
  - □ Writer
  - IOException
  - ....

#### Habe ich da Stream gehört?



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

BufferedReader readLine — Gut für Textdateien



```
// ! Absoluter Pfad ist nicht empfehlenswert !
  File myFile = new
   → File("C:\\Users\\Marc\\Daten\\TestVerzeichnis\\TextDatei.txt"):
  FileReader fileReader = new FileReader(mvFile):
  // Relativer Pfad zum Programm
  FileReader fileReader = new FileReader("TextDatei.txt"):
3
  BufferedReader bufferedReader = new BufferedReader(fileReader):
5
  String firstLine = bufferedReader.readLine():
  bufferedReader.close():
```

```
// Relativer Pfad zum Programm
FileWriter fileWriter = new FileWriter("TextDatei.txt");

BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);

bufferedWriter.write("First line of the document!\n");

bufferedWriter.close();
```

BufferedWriter write - Gut für Textdateien



```
// Relativer Pfad zum Programm
   FileWriter fileWriter = new FileWriter("TextDatei.txt"):
   BufferedWriter writer = new BufferedWriter(fileWriter):
   writer.write(String.valueOf(true)):
   writer.write(",")
   writer.write(String.valueOf(5.2)):
   writer.write(".")
   writer.write(String.valueOf(10)):
10
   writer.close():
```

## File-IO mit Streams - Files.lines



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

#### Das steht heute auf dem Plan



File-IO

IOException?
Das Problem
try-with-resources

String

#### **IOException?**



```
try {
       FileReader fileReader = new FileReader("TextDatei.txt");
       BufferedReader bufferedReader = new
        → BufferedReader(fileReader);
5
       String firstLine = bufferedReader.readLine();
       // 111
       bufferedReader.close();
10
   catch(IOException e) {
       e.printStackTrace();
```

### **IOException?**

**Das Problem** 



```
FileReader reader = null:
try {
    reader = new FileReader("TextDatei.txt");
    // ....
catch(IOException e) {
    e.printStackTrace():
finally {
   // Close Reader
```

### **IOException?**

**Das Problem** 



```
FileReader reader = null;
. . . .
finally {
    if(reader != null) {
        try {
             reader.close();
        catch(IOException e) {
            e.printStackTrace();
```

## IOException? try-with-resources



```
try(FileReader fileReader = new FileReader("TextDatei.txt");
    RufferedReader bufferedReader = new
    → BufferedReader(fileReader)) {
    String firstLine = bufferedReader.readLine();
catch(IOException e) {
   e.printStackTrace():
```

# IOException? try-with-resources



```
try(BufferedReader bufferedReader = createANewReader()) {
    String firstLine = bufferedReader.readLine();
}
catch(IOException e) {
    e.printStackTrace();
}
```

#### Das steht heute auf dem Plan



File-IO

IOException?

String split contains indexOf substring

# String split



```
String text = "Dieser Text ist toll!";

String[] split = text.split(" ");

System.out.println(Arrays.toString(split));

// -> [Dieser, Text, ist, toll!]
```

# String split



```
String text = "A,B,C,";

String[] split = text.split(",");

System.out.println(Arrays.toString(split));
// -> [A, B, C]
```

## String contains



```
String text = "A,B,C,";

System.out.println(text.contains("B"));
// -> true
System.out.println(text.contains("Z"));
// -> false
```

## String indexOf



```
String text = "A,B,C,";

System.out.println(text.indexOf("B"));
// -> 2
System.out.println(text.indexOf("Z"));
// -> -1
System.out.println(text.indexOf(","));
// -> 1
```

# String substring



```
String text = "0123456789";

String a = text.substring(1);
// -> 123456789

String b = text.substring(2);
// -> 23456789

String c = text.substring(2, 5);
// -> 234
```

### Das steht heute auf dem Plan



File-IO

IOException?

String



```
String text = "257";

// NumberFormatException?
int num = Integer.parseInt(text);

String other = "25.112";

// NumberFormatException?
double otherNum = Double.parseDouble(other);
```



```
int num = 257;

String text = String.valueOf(num);

double otherNum = 25.112;

String other = String.valueOf(otherNum);
```

# **Live-Coding!**