Propädeutikum:
Programmierung in der Bioinformatik
I/O Streams und Exceptions

Thomas Mauermeier
10.12.2019
Ludwig-Maximilians-Universität München

Datei einlesen: Bisheriger Approach

```java
public static void main(String[] args) throws IOException {
    ArrayList<String> lines = new ArrayList<>();
    String filename = "test.fasta";
    Path path = FileSystems.getDefault().getPath(filename);
    lines = (ArrayList<String>) Files.readAllLines(path);
}
```

Liest *alle* Zeilen auf einmal ein – funktioniert wunderbar bei **kleinen** Files

→ **Problem:** Was, wenn ich einen riesigen File einlesen will, oder nur bestimmte Teile?

```
#!genome-build GRCh38.p7
#!genome-version GRCh38
#!genome-date 2013-12
#!genome-build-accession NCBI:GCA_000001405.22
#!genebuild-last-updated 2016-06
1	havana	gene	11869	14409	.	+	.	gene_id "ENSG00000223972"; gene_version "5"; gene_name "DDX11L1"; gene_source "havana"; gene_biotype "transcribed_unprocesse
1	havana	transcript	11869	14409	.	+	.	gene_id "ENSG00000223972"; gene_version "5"; transcript_id "ENST00000456328"; transcript_version "2"; gene_name "DD
1	havana	exon	11869	12227	.	+	.	gene_id "ENSG00000223972"; gene_version "5"; transcript_id "ENST00000456328"; transcript_version "2"; exon_number "1"; gene_
1	havana	exon	12613	12721	.	+	.	gene_id "ENSG00000223972"; gene_version "5"; transcript_id "ENST00000456328"; transcript_version "2"; exon_number "2"; gene_
1	havana	exon	13221	14409	.	+	.	gene_id "ENSG00000223972"; gene_version "5"; transcript_id "ENST00000456328"; transcript_version "2"; exon_number "3"; gene_
1	havana	transcript	12010	13670	.	+	.	gene_id "ENSG00000223972"; gene_version "5"; transcript_id "ENST00000450305"; transcript_version "2"; gene_name "DD
1	havana	exon	12010	12057	.	+	.	gene_id "ENSG00000223972"; gene_version "5"; transcript_id "ENST00000450305"; transcript_version "2"; exon_number "1"; gene_
1	havana	exon	12179	12227	.	+	.	gene_id "ENSG00000223972"; gene_version "5"; transcript_id "ENST00000450305"; transcript_version "2"; exon_number "2"; gene_
1	havana	exon	12613	12697	.	+	.	gene_id "ENSG00000223972"; gene_version "5"; transcript_id "ENST00000450305"; transcript_version "2"; exon_number "3"; gene_
1	havana	exon	12975	13052	.	+	.	gene_id "ENSG00000223972"; gene_version "5"; transcript_id "ENST00000450305"; transcript_version "2"; exon_number "4"; gene_
1	havana	exon	13221	13374	.	+	.	gene_id "ENSG00000223972"; gene_version "5"; transcript_id "ENST00000450305"; transcript_version "2"; exon_number "5"; gene_
1	havana	exon	13453	13670	.	+	.	gene_id "ENSG00000223972"; gene_version "5"; transcript_id "ENST00000450305"; transcript_version "2"; exon_number "6"; gene_
1	havana	gene	14404	29570	.	-	.	gene_id "ENSG00000227232"; gene_version "5"; gene_name "WASH7P"; gene_source "havana"; gene_biotype "unprocessed_pseudogene
1	havana	transcript	14404	29570	.	-	.	gene_id "ENSG00000227232"; gene_version "5"; transcript_id "ENST00000488147"; transcript_version "1"; gene_name "WA
1	havana	exon	29534	29570	.	-	.	gene_id "ENSG00000227232"; gene_version "5"; transcript_id "ENST00000488147"; transcript_version "1"; exon_number "1"; gene_
1	havana	exon	24738	24891	.	-	.	gene_id "ENSG00000227232"; gene_version "5"; transcript_id "ENST00000488147"; transcript_version "1"; exon_number "2"; gene_
1	havana	exon	18268	18366	.	-	.	gene_id "ENSG00000227232"; gene_version "5"; transcript_id "ENST00000488147"; transcript_version "1"; exon_number "3"; gene_
1	havana	exon	17915	18061	.	-	.	gene_id "ENSG00000227232"; gene_version "5"; transcript_id "ENST00000488147"; transcript_version "1"; exon_number "4"; gene_
1	havana	exon	17606	17742	.	-	.	gene_id "ENSG00000227232"; gene_version "5"; transcript_id "ENST00000488147"; transcript_version "1"; exon_number "5"; gene_
1	havana	exon	17233	17368	.	-	.	gene_id "ENSG00000227232"; gene_version "5"; transcript_id "ENST00000488147"; transcript_version "1"; exon_number "6"; gene_
1	havana	exon	16858	17055	.	-	.	gene_id "ENSG00000227232"; gene_version "5"; transcript_id "ENST00000488147"; transcript_version "1"; exon_number "7"; gene_
1	havana	exon	16607	16765	.	-	.	gene_id "ENSG00000227232"; gene_version "5"; transcript_id "ENST00000488147"; transcript_version "1"; exon_number "8"; gene_
1	havana	exon	15796	15947	.	-	.	gene_id "ENSG00000227232"; gene_version "5"; transcript_id "ENST00000488147"; transcript_version "1"; exon_number "9"; gene_
1	havana	exon	15005	15038	.	-	.	gene_id "ENSG00000227232"; gene_version "5"; transcript_id "ENST00000488147"; transcript_version "1"; exon_number "10"; gen
1	havana	exon	14404	14501	.	-	.	gene_id "ENSG00000227232"; gene_version "5"; transcript_id "ENST00000488147"; transcript_version "1"; exon_number "11"; gen
1	mirbase	gene	17369	17436	.	-	.	gene_id "ENSG00000278267"; gene_version "1"; gene_name "MIR6859-1"; gene_source "mirbase"; gene_biotype "miRNA";
1	mirbase	transcript	17369	17436	.	-	.	gene_id "ENSG00000278267"; gene_version "1"; transcript_id "ENST00000619216"; transcript_version "1"; gene_name "MI
1	mirbase	exon	17369	17436	.	-	.	gene_id "ENSG00000278267"; gene_version "1"; transcript_id "ENST00000619216"; transcript_version "1"; exon_number "1"; gene_
1	ensembl_havana	gene	29554	31109	.	+	.	gene_id "ENSG00000243485"; gene_version "4"; gene_name "MIR1302-2"; gene_source "ensembl_havana"; gene_biotype "lin
1	havana	transcript	29554	31097	.	+	.	gene_id "ENSG00000243485"; gene_version "4"; transcript_id "ENST00000473358"; transcript_version "1"; gene_name "MI
1	havana	exon	29554	30039	.	+	.	gene_id "ENSG00000243485"; gene_version "4"; transcript_id "ENST00000473358"; transcript_version "1"; exon_number "1"; gene_
1	havana	exon	30564	30667	.	+	.	gene_id "ENSG00000243485"; gene_version "4"; transcript_id "ENST00000473358"; transcript_version "1"; exon_number "2"; gene_
1	havana	exon	30976	31097	.	+	.	gene_id "ENSG00000243485"; gene_version "4"; transcript_id "ENST00000473358"; transcript_version "1"; exon_number "3"; gene_
1	havana	transcript	30267	31109	.	+	.	gene_id "ENSG00000243485"; gene_version "4"; transcript_id "ENST00000469289"; transcript_version "1"; gene_name "MI
1	havana	exon	30267	30667	.	+	.	gene_id "ENSG00000243485"; gene_version "4"; transcript_id "ENST00000469289"; transcript_version "1"; exon_number "1"; gene_
1	havana	exon	30976	31109	.	+	.	gene_id "ENSG00000243485"; gene_version "4"; transcript_id "ENST00000469289"; transcript_version "1"; exon_number "2"; gene_
1	ensembl	transcript	30366	30503	.	+	.	gene_id "ENSG00000243485"; gene_version "4"; transcript_id "ENST00000607096"; transcript_version "1"; gene_name "MI
1	ensembl	exon	30366	30503	.	+	.	gene_id "ENSG00000243485"; gene_version "4"; transcript_id "ENST00000607096"; transcript_version "1"; exon_number "1"; gene_
1	havana	gene	34554	36081	.	-	.	gene_id "ENSG00000237613"; gene_version "2"; gene_name "FAM138A"; gene_source "havana"; gene_biotype "lincRNA"; havana_gene
1	havana	transcript	34554	36081	.	-	.	gene_id "ENSG00000237613"; gene_version "2"; transcript_id "ENST00000417324"; transcript_version "1"; gene_name "FA
1	havana	exon	35721	36081	.	-	.	gene_id "ENSG00000237613"; gene_version "2"; transcript_id "ENST00000417324"; transcript_version "1"; exon_number "1"; gene_
1	havana	exon	35277	35481	.	-	.	gene_id "ENSG00000237613"; gene_version "2"; transcript_id "ENST00000417324"; transcript_version "1"; exon_number "2"; gene_
1	havana	exon	34554	35174	.	-	.	gene_id "ENSG00000237613"; gene_version "2"; transcript_id "ENST00000417324"; transcript_version "1"; exon_number "3"; gene_
1	havana	transcript	35245	36073	.	-	.	gene_id "ENSG00000237613"; gene_version "2"; transcript_id "ENST00000461467"; transcript_version "1"; gene_name "FA
1	havana	exon	35721	36073	.	-	.	gene_id "ENSG00000237613"; gene_version "2"; transcript_id "ENST00000461467"; transcript_version "1"; exon_number "1"; gene_
1	havana	exon	35245	35481	.	-	.	gene_id "ENSG00000237613"; gene_version "2"; transcript_id "ENST00000461467"; transcript_version "1"; exon_number "2"; gene_
1	havana	gene	52473	53312	.	+	.	gene_id "ENSG00000268020"; gene_version "3"; gene_name "OR4F29"; gene_source "havana"; gene_biotype "unprocessed_pseudogene
1	havana	transcript	52473	53312	.	+	.	gene_id "ENSG00000268020"; gene_version "3"; transcript_id "ENST00000606857"; transcript_version "1"; gene_name "OR
1	havana	exon	52473	53312	.	+	.	gene_id "ENSG00000268020"; gene_version "3"; transcript_id "ENST00000606857"; transcript_version "1"; exon_number "1"; gene_
```

z.B.: Wir wollen nur Zeilen einlesen die ein Transcript beschreiben

Problem: File hat 2’575’499 Zeilen und ist 1,3 GB groß

→ Nicht sehr effizient einfach mal pauschal alles einzulesen

Datei einlesen mit einem BufferedReader

```
public static void main(String[] args) throws IOException {
    String filename = "test.gtf";
    ArrayList<String> lines = new ArrayList<>();
    BufferedReader reader = new BufferedReader(new FileReader(filename));
    String line;
    while ((line = reader.readLine()) != null) {
        if (line.contains("transcript")) {
           lines.add(line);
        }
    }
}
```

Bis hier noch nichts neues oder zwingend notwendiges für **`BufferedReader`**:

- File aus dem wir lesen wollen deklarieren
- ArrayList in der wir unsere Zeilen speichern

Datei einlesen mit einem BufferedReader

```
public static void main(String[] args) throws IOException {
    String filename = "test.gtf";
    ArrayList<String> lines = new ArrayList<>();
    BufferedReader reader = new BufferedReader(new FileReader(filename));
    String line;
    while ((line = reader.readLine()) != null) {
        if (line.contains("transcript")) {
            lines.add(line);
        }
    }
}
```

Erstellen eines **`BufferedReader`**-Objekts

- Ermöglicht mir Zugriff auf die Datei die ich lesen möchte (z.B. **`reader.readLine()`**)
- Konstruktor braucht einen **`FileReader`** als Parameter, und diesem **`FileReader`** geben wir File
  - **`FileReader`** = Ungepufferte Variante eines Readers (→ “liest nicht so effizient”)

Datei einlesen mit einem BufferedReader

```
public static void main(String[] args) throws IOException {
    String filename = "test.gtf";
    ArrayList<String> lines = new ArrayList<>();
    BufferedReader reader = new BufferedReader(new FileReader(filename));
    String line;
    while ((line = reader.readLine()) != null) {
        if (line.contains("transcript")) {
            lines.add(line);
        }
    }
}
```

Inhalt der Datei:

Position des Readers im File

1 havana gene
1 havana transcript
1 havana exon
1 havana exon
1 havana gene
1 havana transcript
null

`line = reader.readLine()`

Zwischenspeicher befüllen bzw. Zeile einlesen

1 havana gene

Inhalt von Variable line

Datei einlesen mit einem BufferedReader

```
public static void main(String[] args) throws IOException {
    String filename = "test.gtf";
    ArrayList<String> lines = new ArrayList<>();
    BufferedReader reader = new BufferedReader(new FileReader(filename));
    String line;
    while ((line = reader.readLine()) != null) {
        if (line.contains("transcript")) {
            lines.add(line);
        }
    }
}
```

Inhalt der Datei:

Position des Readers im File

1 havana gene
1 havana transcript
1 havana exon
1 havana exon
1 havana gene
1 havana transcript
null

`line = reader.readLine()`

Zwischenspeicher befüllen bzw. Zeile einlesen

1 havana transcript

Inhalt von Variable line

Datei einlesen mit einem BufferedReader

```
public static void main(String[] args) throws IOException {
    String filename = "test.gtf";
    ArrayList<String> lines = new ArrayList<>();
    BufferedReader reader = new BufferedReader(new FileReader(filename));
    String line;
    while ((line = reader.readLine()) != null) {
        if (line.contains("transcript")) {
            lines.add(line);
        }
    }
}
```

Inhalt der Datei:

Position des Readers im File

1 havana gene
1 havana transcript
1 havana exon
1 havana exon
1 havana gene
1 havana transcript
null

→ `line = reader.readLine()`

Zwischenspeicher befüllen bzw. Zeile einlesen

→ 1 havana exon

Inhalt von Variable line

Datei einlesen mit einem BufferedReader

```
public static void main(String[] args) throws IOException {
    String filename = "test.gtf";
    ArrayList<String> lines = new ArrayList<>();
    BufferedReader reader = new BufferedReader(new FileReader(filename));
    String line;
    while ((line = reader.readLine()) != null) {
        if (line.contains("transcript")) {
            lines.add(line);
        }
    }
}
```

Inhalt der Datei:

Position des Readers im File

1 havana gene
1 havana transcript
1 havana exon
1 havana exon
1 havana gene
1 havana transcript
null

`line = reader.readLine()`

Zwischenspeicher befüllen bzw. Zeile einlesen

1 havana exon

Inhalt von Variable line

Datei einlesen mit einem BufferedReader

```
public static void main(String[] args) throws IOException {
    String filename = "test.gtf";
    ArrayList<String> lines = new ArrayList<>();
    BufferedReader reader = new BufferedReader(new FileReader(filename));
    String line;
    while ((line = reader.readLine()) != null) {
        if (line.contains("transcript")) {
            lines.add(line);
        }
    }
}
```

Inhalt der Datei:

Position des Readers im File

1 havana gene
1 havana transcript
1 havana exon
1 havana exon
1 havana gene
1 havana transcript
null

`line = reader.readLine()`

Zwischenspeicher befüllen bzw. Zeile einlesen

1 havana gene

Inhalt von Variable line

Datei einlesen mit einem BufferedReader

```
public static void main(String[] args) throws IOException {
    String filename = "test.gtf";
    ArrayList<String> lines = new ArrayList<>();
    BufferedReader reader = new BufferedReader(new FileReader(filename));
    String line;
    while ((line = reader.readLine()) != null) {
        if (line.contains("transcript")) {
            lines.add(line);
        }
    }
}
```

Inhalt der Datei:

Position des Readers im File

1 havana gene
1 havana transcript
1 havana exon
1 havana exon
1 havana gene
1 havana transcript
null

`line = reader.readLine()`

Zwischenspeicher befüllen bzw. Zeile einlesen

1 havana transcript

Inhalt von Variable line

Datei einlesen mit einem BufferedReader

```
public static void main(String[] args) throws IOException {
    String filename = "test.gtf";
    ArrayList<String> lines = new ArrayList<>();
    BufferedReader reader = new BufferedReader(new FileReader(filename));
    String line;
    while ((line = reader.readLine()) != null) {
        if (line.contains("transcript")) {
            lines.add(line);
        }
    }
}
```

Inhalt der Datei:

1 havana gene
1 havana transcript
1 havana exon
1 havana exon
1 havana gene
1 havana transcript
null

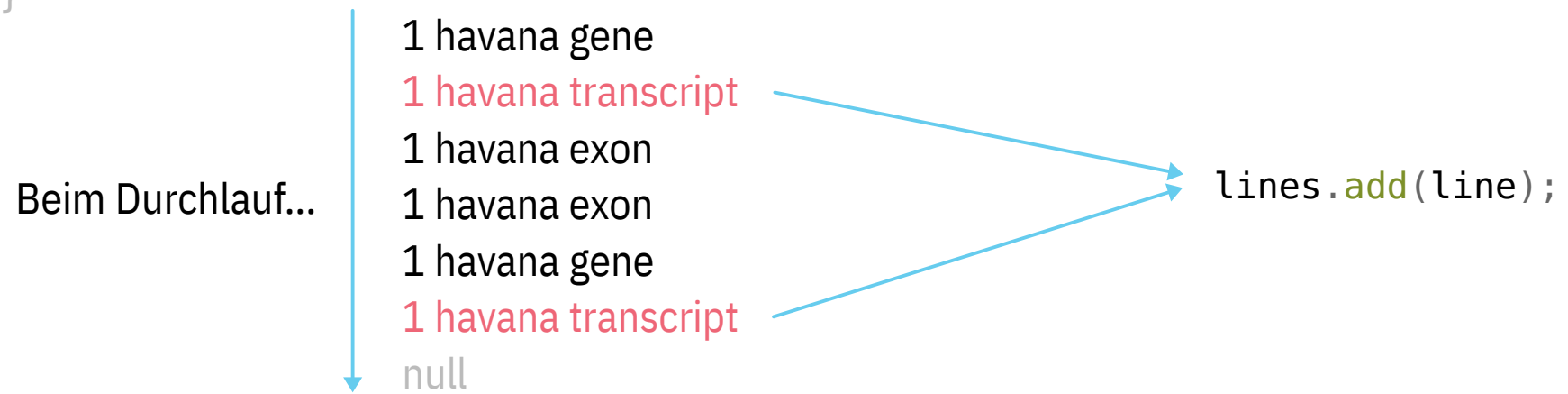
EOF!
(End of File)

→ **Abbruch der while-Schleife!**

Achtung: Abschließendes schließen des Readers fehlt hier, mehr dazu später

Datei einlesen mit einem BufferedReader

```
public static void main(String[] args) throws IOException {
    String filename = "test.gtf";
    ArrayList<String> lines = new ArrayList<>();
    BufferedReader reader = new BufferedReader(new FileReader(filename));
    String line;
    while ((line = reader.readLine()) != null) {
        if (line.contains("transcript")) {
            lines.add(line);
        }
    }
}
```



Datei einlesen mit einem BufferedReader

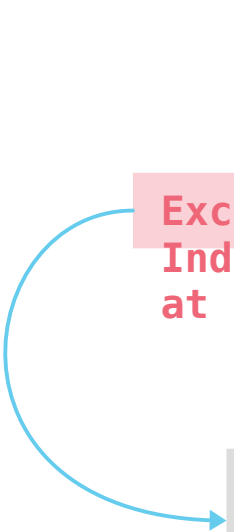
```
public static void main(String[] args) throws IOException {
    String filename = "test.gtf";
    ArrayList<String> lines = new ArrayList<>();
    BufferedReader reader = new BufferedReader(new FileReader(filename));
    String line;
    while ((line = reader.readLine()) != null) {
        if (line.contains("transcript")) {
           lines.add(line);
        }
    }
}
```

Teil *innerhalb* der while-Schleife wird dann genutzt um jeweilige Line “anzuschauen”
Häufig macht man hier folgende Dinge:

- Enthält Zeile das was ich will?
- Muss ich die Zeile noch irgendwie aufspalten?
- Will ich die Zeile irgendwo abspeichern, und wie?

Exceptions

```
int[] zahlenArray = new int[10];
zahlenArray[12] = 7;
```



```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:
Index 12 out of bounds for length 10
at ReadAllLinesTest.main(ReadAllLinesTest.java:27)
```

Exceptions sind (wie fast alles andere in Java) Objekte!
→ Man kann also Fehler im Code irgendwie verwenden, aber **wann** und **wie**?

Wann muss ich Exceptions behandeln?

- Checked Exceptions
  - Dinge die ein gut geschriebenes Programm abfangen können sollte
  - **müssen** behandelt werden
    - Klassen wie BufferedReader, BufferedWriter **verlangen** dass Exception behandelt wird
- Runtime Exceptions
  - Logische Fehler im Code, wie z.B. Zugriff auf nicht existierende Felder in Array
  - muss nicht behandelt werden
- Errors
  - Fehler die außerhalb des Codes entstehen, wie z.B. Hardwareprobleme
  - muss nicht behandelt werden

Woher weiß ich was eine Exception wirft?

Kurzfassung: Lernt *bitte, bitte, bitte* die Java API zu lesen!

Langfassung: In der Java API steht bei allen Methoden dabei ob und welche Exception sie werfen

z.B. Auszug zur Methode **`readLine()`** der Klasse **`BufferedReader`**:

readLine

```
public String readLine() throws IOException
```

Reads a line of text. A line is considered to be terminated by any one of a line feed ('\n'), a carriage return ('\r'), a c

Returns:

A String containing the contents of the line, not including any line-termination characters, or null if the end of the

Throws:

IOException - If an I/O error occurs

See Also:

Files.readAllLines(java.nio.file.Path, java.nio.charset.Charset)

Wie muss ich Exceptions behandeln?

```
try {
    // Code der Exception werfen kann
    // z.B. ein BufferedReader
} catch (ExceptionTyp e) {
    // Was tun wenn Exception vom Typ
    // "ExceptionTyp" eintritt?
} finally {
    // Teil der immer ausgeführt wird
    // egal ob es Exception gab oder nicht
}
```

try-catch Block

- zum tatsächlichen Behandeln der Exception im catch-Block

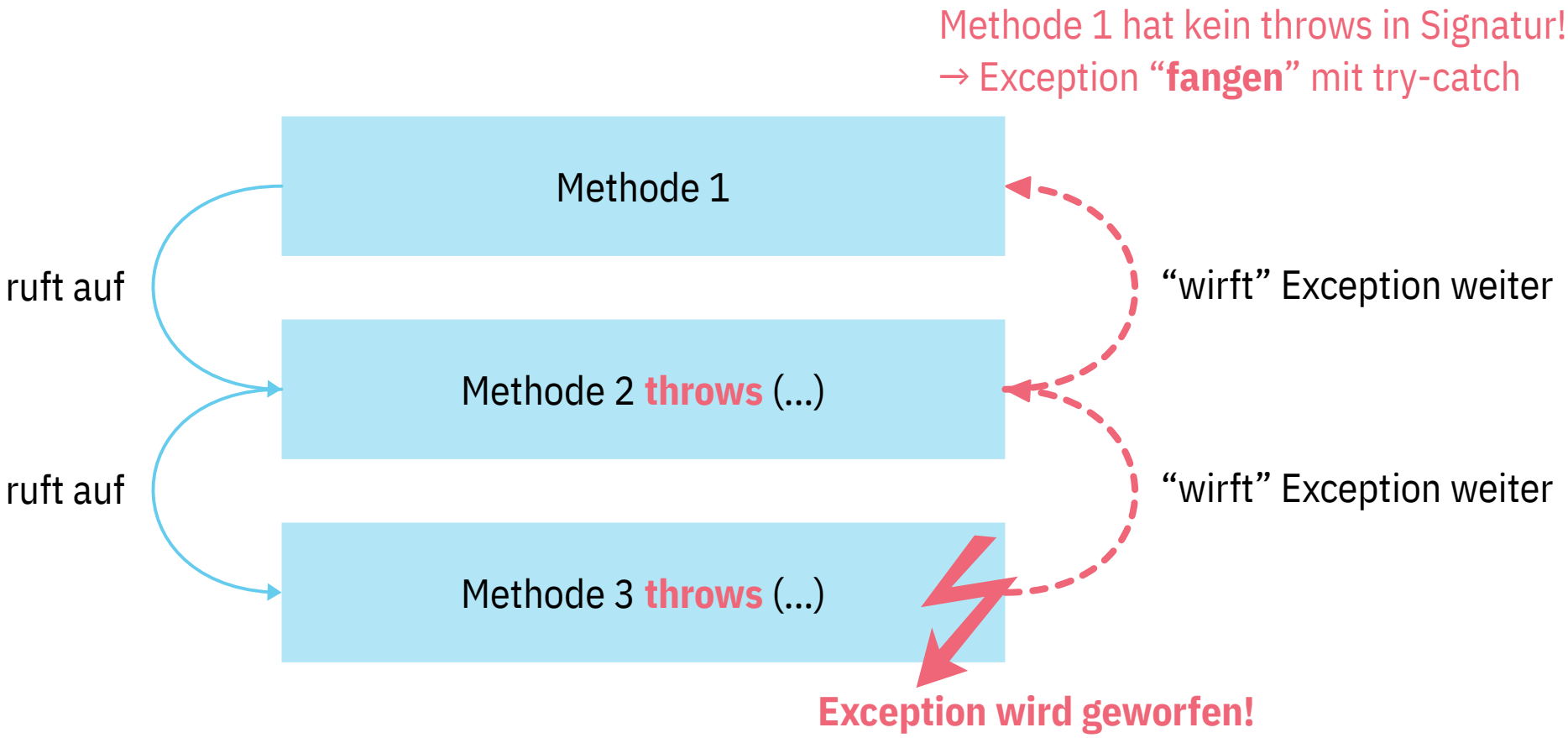
```
public void methode() throws ExceptionTyp {
    // Inhalt der Methode, mit
    // Code der Exception werfen kann
}
```

throws Spezifikation

- zum “weitergeben” des Fehlers an die aufrufende Methode

Ausführlicher: http://openbook.rheinwerk-verlag.de/javainsel/07_001.html#u7

try-catch und throws im Call-Stack



BufferedReader Exceptions behandeln

```
String filename = "test.gtf";
ArrayList<String> lines = new ArrayList<>();
BufferedReader reader = null;
    try {
        reader = new BufferedReader(new FileReader(filename));
        String line;
        while ((line = reader.readLine()) != null) {
            if (line.contains("transcript")) lines.add(line);
        }
    } catch (FileNotFoundException e) {
        System.out.println("Datei existiert nicht!");
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (reader != null) {
            try { reader.close(); }
            catch (IOException e) { e.printStackTrace(); }
        }
    }
```

try-Block

catch-Blöcke

finally-Block

Achtung: Reader/Writer müssen nach Verwendung immer geschlossen (**`reader.close()`**) werden!

BufferedReader Exceptions einfach behandeln

```
String filename = "test.gtf";
ArrayList<String> lines = new ArrayList<>();
    try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {
        String line;
        while ((line = reader.readLine()) != null) {
            if (line.contains("transcript")) lines.add(line);
        }
    } catch (FileNotFoundException e) {
        System.out.println("Datei existiert nicht!");
    } catch (IOException e) {
        e.printStackTrace();
    }
```

try-with-resources:

- Erstellen des Readers *vor* dem **`try`**-Block in Klammern
- Reader schließen nun nicht mehr notwendig – passiert automatisch!
- Daher kann der komplizierte **`finally`**-Block ausgelassen werden

Ausführlicher: http://openbook.rheinwerk-verlag.de/javainsel/07_006.html#u7.6

Datei schreiben mit einem BufferedWriter

```
String outputFile = "output.gtf";
    try (BufferedWriter writer = new BufferedWriter(new FileWriter(outputFile))) {
        for (String s : lines) {
            writer.write(s + '\n');
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
```

Läuft analog zu **`BufferedReader`**:

- Erstellen eines **`BufferedWriter`**, der mir ermöglicht in Datei zu schreiben
- Statt mit einer **`read()`**-Methode zu lesen, schreiben wir mit **`write(String s)`**
- Exceptions müssen auch beim Schreiben behandelt werden

Links

- Zu Exceptions:
  - Oracle Java Tutorials: Exceptions
  - Java ist auch eine Insel: Ausnahmen müssen sein
- Zum Lesen/Schreiben:
  - Oracle Java Tutorials: Basic I/O
  - Java ist auch eine Insel: Einführung in Dateien und Datenströme