

252-0027

**Einführung in die Programmierung
Übungen**

Compile- & Runtime-Fehler, Arrays

**Henrik Pätzold
Departement Informatik
ETH Zürich**

Heutiger Plan

- Theorie
 - Erinnerung: Erst Pullen, dann Pushen (ansonsten gibt es Merge-Conflicts)
 - Compile-Fehler & Exceptions (contd.)
 - Casting (Basics)
 - Aufgaben zu Compile-Fehlern und Runtime-Exceptions
 - String-Escaping (kurz)
- Praxis
 - Kollektives Coding mit Arrays
- Vorbesprechung - Übung 3

Arithmetische Compiler-Fehler und Exceptions

- Compiler-Fehler treten auf, bevor die Laufzeit beginnt
 - Häufigste (arith.) Compiler-Fehler: Wenn zwei inkompatible Datentypen in einer Operation verwendet werden.
 - *Beispiel*: `int x = 5.0 + 2, "hallo"*3`
- Exceptions treten zur Laufzeit auf
 - Häufigste arithmetische Exception: Division durch Null
 - Der Compiler überprüft nur die Syntax, nicht die Funktionsweise des Programms (z. B. logische Fehler werden nicht erkannt).

Erkennen, ob Compile-Fehler oder Exception


Zuerst Compile-Fehler,...

- Typenkompatibilität zwischen Operationen überprüfen
 - Compiler-Brille (später mehr)
 - Ist Type-Casting korrekt verwendet worden?
- Syntax überprüfen
 - Keine Klammern, Semikolons vergessen?
 - Werden Methoden richtig aufgerufen?

... dann Exceptions

- Wir überprüfen das Programm auf Logikfehler
 - **ArithmeticException**: Tritt auf bei fehlerhaften mathematischen Operationen (z.B. Division durch Null).
 - **NullPointerException**: Entsteht, wenn auf ein *null*-Objekt zugegriffen wird.
 - **ClassCastException**: Wird geworfen bei einem ungültigen Cast zwischen inkompatiblen Objekten.
 - **ArrayIndexOutOfBoundsException**: Passiert, wenn auf einen ungültigen Index eines Arrays zugegriffen wird.
 - **NumberFormatException**: Tritt auf, wenn versucht wird, einen String in eine Zahl umzuwandeln, der kein korrektes Zahlenformat hat.

Exceptions können seeeehr spezifisch werden



```
1 // Eigene Exception
2 class OverheatedException extends Exception {
3     public OverheatedException(String message) {
4         super(message);
5     }
6 }
7
8 class Machine {
9     private int temperature;
10
11     public Machine(int temperature) {
12         this.temperature = temperature;
13     }
14
15     public void checkTemperature() throws OverheatedException {
16         if (temperature > 100) {
17             throw new OverheatedException("Machine is overheated!");
18         }
19     }
20 }
```


Beispiel 1




```
1 public class MyClass {  
2     public static void main(String[] args){  
3         int a = 0;  
4         int b = 5;  
5         System.out.println((a > 0) && (b / a > 1));  
6     }  
7 }
```

Beispiel 1 – Das sieht der Compiler

```
1 public class MyClass {  
2     public static void main(String[] args){  
3         int a = int;  
4         int b = int;  
5         System.out.println((int > int) && (int / int > int));  
6     }  
7 }
```

Beispiel 1



```
1 public class MyClass {  
2     public static void main(String[] args){  
3         int a = 0;  
4         int b = 5;  
5         System.out.println((a > 0) && (b / a > 1));  
6     }  
7 }
```

A red bracket is drawn under the expression `(a > 0)` in line 5, with the word **False** written in red text below it.

False

Beispiel 1 – Ohne Short-Circuiting



```
1 public class MyClass {  
2     public static void main(String[] args){  
3         int a = 0;  
4         int b = 5;  
5         System.out.println((b != 0) && (b / a > 1));  
6     }  
7 }
```

True Exception

Casting (später mehr bei Vererbung)

Basics

- **Wieso:** Casting erlaubt es, einen Datentyp in einen anderen umzuwandeln, wenn die Typen kompatibel sind.
- **Upcasting:** Von einem spezifischen Typ (Unterklasse) zu einem allgemeinen Typ (Oberklasse).
 - Automatisch durchführbar.
- **Downcasting:** Von einem allgemeinen Typ (Oberklasse) zu einem spezifischen Typ (Unterklasse).
 - Muss explizit angegeben werden und kann zu `ClassCastException` führen.

int → **32 bits** **long** → **64 bits** **double**
64 bits

Aufgabe 1



```
1 int x = 10  
2 double y = 5.0;  
3 int z = x + y;
```

Compile-Fehler



Exception



Aufgabe 2



```
1 int x = 10;  
2 int y = 5;  
3 double result = (double x + y;
```

Compile-Fehler



Exception



Aufgabe 3



```
1 String num = "123";  
2 int number = (int) num;  
3 double result = 2 * number;
```

Compile-Fehler



Exception



Aufgabe 4



```
1 double d = 10.9;  
2 int x = (int) d;  
3 System.out.println("x is: " + x);
```

Compile-Fehler

☐

Exception

☐

Weder, noch!

Aufgabe 5



```
1 int x = 0;
2 int y = 5;
3 Integer res;
4
5 for (int i = x; i < y; i++) {
6     res = res / (x - y);
7     y--;
8 }
```

Compile-Fehler



Exception



Aufgabe 6



```
1 double x = 10.5;  
2 int y = (int) (x * "2");  
3 int z = x
```

Compile-Fehler



Exception



Aufgabe 7



```
1 Integer a = null;  
2 int b = 5;  
3 int result = a + b;
```

Compile-Fehler



Exception



Aufgabe 8



```
1 int a = 5;  
2 double b = 2.0;  
3 double result = a / b + a * (b - 1);
```

Compile-Fehler

☐

Exception

☐

Weder, noch!

Escaping in Strings

- Strings in Java sind gekennzeichnet durch `""` (reservierte Symbole)
- Was machen wir, wenn wir ein reserviertes Symbol im String verwenden wollen?
- Wir können es mit `\` (noch ein reserviertes Symbol) "escapen"

"\" \" \" \" \" \" \" \" \" \" \" \"

""\"\"\\\"\\\"\\\"\\\"\\\"\\\"\\\"\\\"\""

Aussengrenzen des Strings
Escape-Charakter
Im Resultat sichtbar



A diagram illustrating string boundaries and escape characters. It features a sequence of vertical bars and diagonal slashes. From left to right: two purple bars, a red diagonal slash, a green bar, a green double quote, five black diagonal slashes, a black double quote, a black backslash, a black double quote, and two purple bars. This sequence represents the visual output of a string containing double quotes and backslashes, where escape sequences are visible.

Aussengrenzen des Strings
Escape-Charakter
Im Resultat sichtbar



Aussengrenzen des Strings

Escape-Charakter

Im Resultat sichtbar



Aussengrenzen des Strings
Escape-Charakter
Im Resultat sichtbar



Aussengrenzen des Strings
Escape-Charakter
Im Resultat sichtbar

""\\""

Aussengrenzen des Strings
Escape-Charakter
Im Resultat sichtbar

Coding

Theorie-ToDo

- ☒ Precedence
- ☒ Short-Circuiting
- ☒ Casting
 - ☒ Generell
 - ☐ Klassen und Vererbung
- ☒ Compiler-Fehler
 - ☒ Generell
 - ☐ Klassen und Vererbung
- ☒ Exceptions
 - ☒ Generell
 - ☐ Klassen und Vererbung
- ☐ Weakest Precondition
- ☐ Loop-Invariante
- ☐ In-Depth EBNF
- ☐ Vererbung & Polymorphismus

Vorbesprechung Übungsblatt 2