

252-0027

**Einführung in die Programmierung
Übungen**

Compile- & Runtime-Fehler, Arrays

**Henrik Pätzold
Departement Informatik
ETH Zürich**

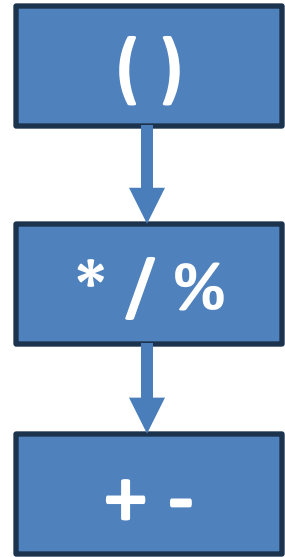
Heutiger Plan

- Nachtrag zu letzter Woche
- Theorie
 - Compile-Fehler & Exceptions (contd.)
 - Aufgaben zu Compile-Fehlern und Runtime-Exceptions
- Praxis
 - Programmieraufgaben
- Vorbesprechung - Übung 3

Nachtrag zu letzter Woche

Operatorpräzedenz

- Operand wird vom Operator mit höherer Rangordnung («precedence») verwendet
- Wenn zwei Operatoren dieselbe Rangordnung haben, dann entscheidet die Assoziativität
- Wenn etwas anderes gewünscht wird: Klammern verwenden!



String-Pooling

11. Lösen Sie die Aufgaben ausschliesslich auf Basis des geschriebenen Codes, ohne dabei mögliche Optimierungen der JVM einzubeziehen.

Compiler-Fehler vs Exceptions


Zuerst Compile-Fehler,...

- Syntax überprüfen
 - Keine Klammern, Semikolons vergessen?
 - Existiert eine Methode mit diesem Namen und dieser Signatur?
- Typenkompatibilität überprüfen
 - Compiler-Brille (später mehr)
 - `int i = 4.5;`
 - **Präzisionsverlust:** Der Compiler beschwert sich, explizites Casting erforderlich.
 - `String s = (String) new Integer(42);`
 - **Kein Vererbungsverhältnis:** Ein Cast zu Laufzeit würde nie funktionieren.

... 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 sehr spezifisch sein...



```
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

Aufgabe 1



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

Compile-Fehler

☐

Exception

☐

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 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 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

☐

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

Compile-Fehler

☐

Exception

☐

Aufgabe 5



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

Compile-Fehler



Exception



Aufgabe 6



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

Compile-Fehler

☐

Exception

☐

Aufgabe 6



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

Compile-Fehler



Exception



Aufgabe 7



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

Compile-Fehler

☐

Exception

☐

Aufgabe 7



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

Compile-Fehler

☐

Exception

☐

Weder, noch!

Programmieraufgabe - WarmUp

1. Implementieren Sie die Methode `Calculations.checksum(int x)`, das heisst die Methode `checksum` in der Klasse `Calculations`. Die Methode nimmt einen Integer `x` als Argument, welcher einen nicht-negativen Wert hat. Die Methode soll die Quersumme von `x` zurückgeben. Sie sollen für diese Aufgabe **keine** Schleife verwenden.

Beispiele

- `checksum(258)` gibt 15 zurück.
- `checksum(49)` gibt 13 zurück.
- `checksum(12)` gibt 3 zurück.

Hinweis: Für einen Integer `a` ist `a % 10` die letzte Ziffer und `a / 10` entfernt die letzte Ziffer. Zum Beispiel `258 % 10` ist 8 und `258 / 10` ist 25.

2. Implementieren Sie die Methode `Calculations.magic7(int a, int b)`. Die Methode gibt einen Boolean zurück. Die Methode soll `true` zurückgeben, wenn einer der Parameter 7 ist oder wenn die Summe oder Differenz der Parameter 7 ist. Ansonsten soll die Methode `false` zurückgeben.

Beispiele

- `magic7(2,5)` gibt `true` zurück.
- `magic7(7,9)` gibt `true` zurück.
- `magic7(5,6)` gibt `false` zurück.

Hinweis: Mit der Funktion `Math.abs(num)` können Sie den absoluten Wert einer Zahl `num` erhalten.

3. Implementieren Sie die Methode `Calculations.fast12(int z)`. Das Argument `z` ist nicht negativ. Die Methode gibt einen Boolean zurück. Die Methode soll `true` zurückgeben, wenn `z` nahe einem Vielfachen von 12 ist. Eine Zahl `x` ist nahe an einer Zahl `y`, wenn eine der Zahlen um maximal 2 grösser oder kleiner ist als die andere Zahl. Ansonsten soll die Methode `false` zurückgeben.


Beispiele

- `fast12(12)` gibt `true` zurück.
- `fast12(14)` gibt `true` zurück.
- `fast12(10)` gibt `true` zurück.
- `fast12(15)` gibt `false` zurück.

Arrays

Arrays (Eindimensional)

- Arrays belegen eine feste Grösse **n** im Speicher, haben daher auch eine feste Länge
- für jeden Eintrag kann für den deklarierten Datentyp ein Wert gespeichert werden
- auf ein spezifisches Element **i** zugreifen können wir mit **arr[i]**
- Indizierung startet bei **0**, geht also bis **n-1**



```
1 public class Main {  
2     public static void main(String[] args){  
3         int[] arr = {3,5,6,1,2,8};  
4         for(int i = 0; i < arr.length; i++){  
5             arr[i] *= 2;  
6         }  
7     }  
8 }
```

2D Arrays

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

[[1,2,3],[4,5,6],[7,8,9]]

Zweidimensionale Arrays

- eine Matrix könnte als Zweidimensionales Array dargestellt werden
- `int[][] arr = new int[n][n];`
definiert ein Array, welches **n** integer-Arrays der Länge **n** speichert (**nxn-Matrix**)
- eine gesamte Zeile erhalten wir also mit `arr[i]` für $0 \leq i < n$
- einen Eintrag erhalten wir mit `arr[i][j]` für $0 \leq i, j < n$
- **Vorsicht:** Die Längen der Arrays könnten unterschiedlich sein!

```
1 public class Main {
2     public static void main(String[] args) {
3         int[][] arr = {{1, 2, 3},{4, 5, 6},{7, 8, 9}};
4         for (int i = 0; i < arr.length; i++) {
5             for (int j = 0; j < arr[i].length; j++) {
6                 arr[i][j] *= 2;
7             }
8         }
9     }
10 }
```

Programmieraufgabe

Vorbesprechung - Übung 3