252-0027

Einführung in die Programmierung

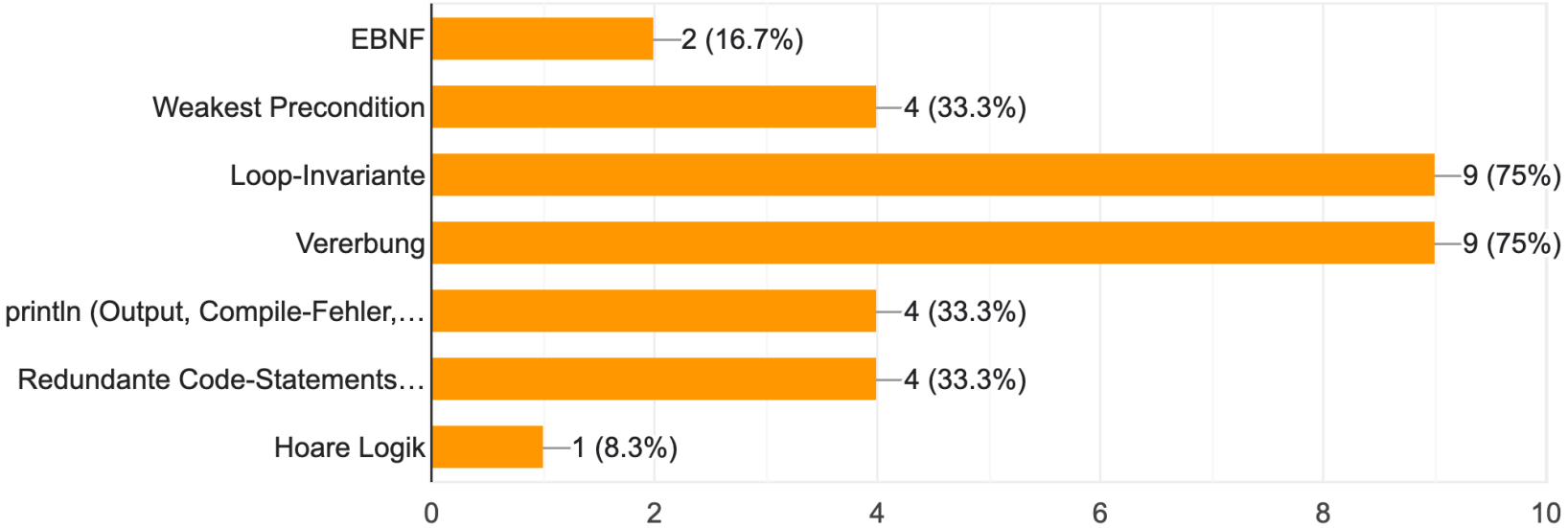
Übungen

Präzedenz und Short-Circuiting

Henrik Pätzold
Departement Informatik
ETH Zürich

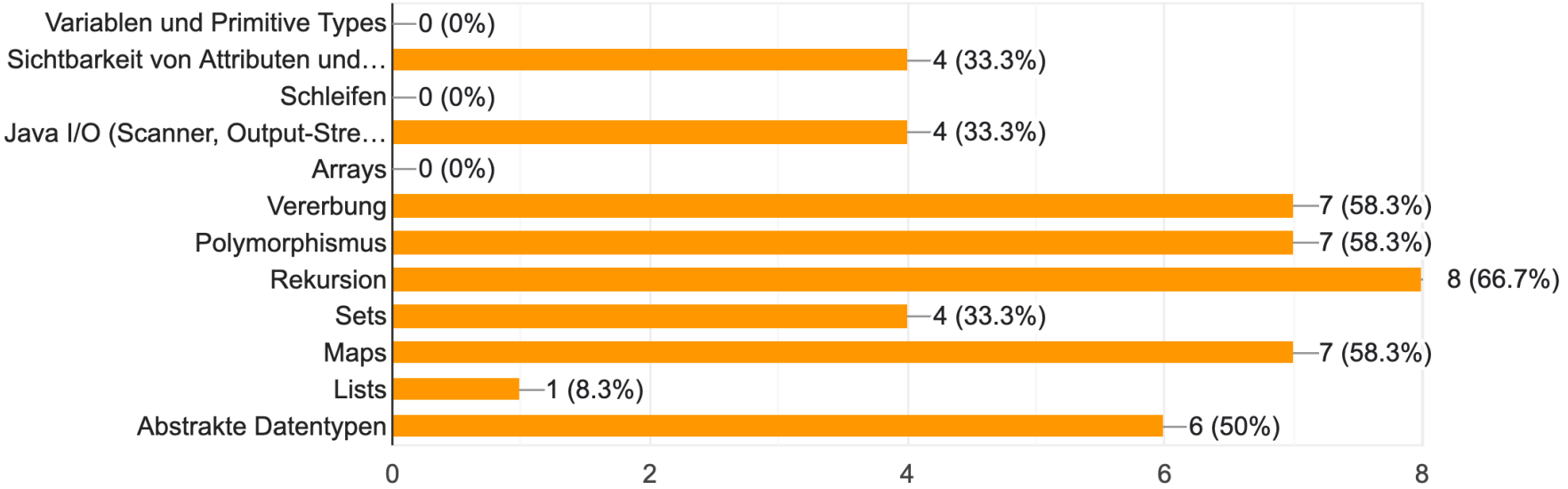
Diese Themen bereiten mir im Theorie-Teil Mühe

12 responses



Diese Themen bereiten mir im Programmieren Mühe

12 responses



Timeline (ungefähr)

1. Einführung
2. Operatorpräzedenz & *println* Statements
3. Compile Fehler & Arrays
4. Call by Value & Recursion
5. Klassen
6. ***Programmierprüfung***
7. Hoare Logic & Weakest Precondition
8. Vererbung I
9. Vererbung II

Aufgabe 1 (3)

Gegeben sei eine Methode `main` in einer Java Klasse.

```
public static void main(String[] args) {
    /* body */
}
```

Die folgenden Anweisungen sollen als "Body" (Rumpf) anstelle des Kommentars `/* body */` eingefügt werden. Geben Sie für jede Anweisung an, was für eine Ausgabe erzeugt wird – entweder was gedruckt wird, oder ob ein Laufzeitfehler auftritt (schreiben Sie "Exception"), oder ob der Compiler einen Fehler feststellt (schreiben Sie "Compile-Fehler"). Achten Sie auf die korrekte Formatierung der verschiedenen Typen, also z.B. 7.0 statt 7 für eine reelle Zahl (`double`).

1. `System.out.println(11 + 16 / 4 * 2 +  (4 + ">") + 4 * 2);`

   ---

2. `System.out.println(20 / 10 % 6 + 3 / 2 + (double) 5 / 4 + 3 / 4);`

   ---

3. `System.out.println( ( 11 % 4 ) >  2  && 9 > (16 / (2 / 4)) && 1 % 8 <  0);`

Checkliste - *println* Aufgaben

- ❑ Operatorpräzedenz
- ❑ Kompilier- und Laufzeitfehler
- ❑ Short-circuiting
- ❑ Escapen von Strings
- ❑ Übung!

Heutiger Plan

- *println*-Statements
  - Operatorpräzedenz
  - Kompilier- und Laufzeitfehler
  - Escapen von Strings
  - Short-circuiting
- Vorbesprechung - Übungen
- Kahoot - EBNF & *println*-Statements

Operator-Präzedenz

Operators	Precedence
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Operator-Präzedenz

Operators	Precedence
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Klammern wie Java

- Nach Präzedenz Klammern setzen
  - ***Linksassoziativität beachten!***
- Einfacher: Klammern zuerst sauber auflösen, bevor zwischen größeren Teiltermen gerechnet wird
  - Von innen nach aussen arbeiten
- Auf den Datentyp zwischen zwei Operationen achten (z. B. String + int , int + double)

int < long < float < double

int < long < float < double

Arithmetische primitive Typen sind nicht kompatibel mit `Boolean`!

int < long < float < double

Arithmetische primitive Typen sind nicht kompatibel mit `Boolean`!

String + primitive = String (String-Konkatenation)

5 + 3 * 2 - 4 / 2 + "" + 6 % 4 + (8 / 2) * 3

{[5 + (3 * 2) – (4 / 2)] + ""} + (6 % 4) + ((8 / 2) * 3)

Wie Java den Ausdruck klammern würde

5 + 3 * 2 - 4 / 2 + "" + 6 % 4 + (8 / 2) * 3

5 + (3 * 2) – (4 / 2) + "" + (6 % 4) + ((8 / 2) * 3))

Wir wollen an einer Prüfung minimalen Aufwand, um zuverlässig zum Ergebnis zu kommen.

5 + 6 - 2 + "" + 2 + 12

(5 + 6) – 2 + "" + 2 + 12

((5 + 6) – 2) + "" + 2 + 12

(((5 + 6) – 2) + "") + 2 + 12

((11– 2) + "") + 2 + 12

(9 + "") + 2 + 12

"9"+ 2 + 12

("9"+ 2) + 12

"92" + 12

"9212"

Checkliste - *println* Aufgaben

- ☑ Operatorpräzedenz
- ❑ Kompilier- und Laufzeitfehler
- ❑ Short-circuiting
- ❑ Escapen von Strings
- ❑ Übung!

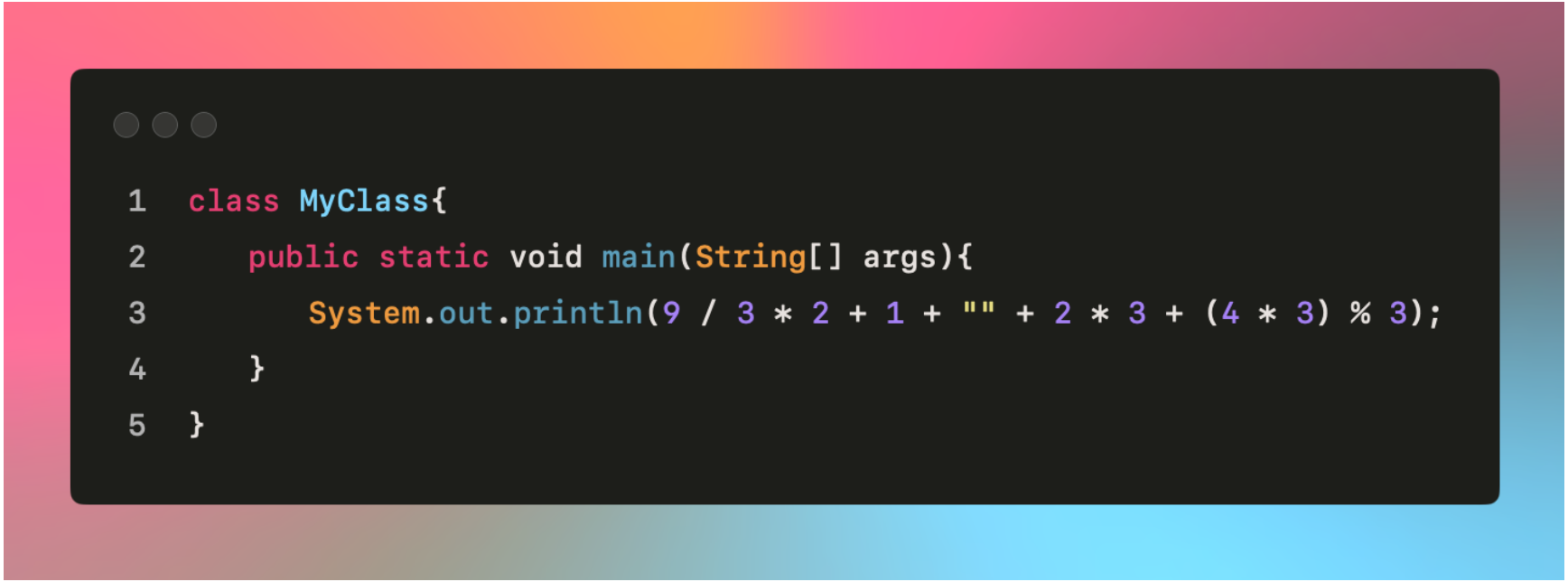
Arithmetische Compiler-Fehler und Exceptions

- Compiler-Fehler treten auf, bevor Ausführung startet
  - Häufigste 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).

Checkliste - *println* Aufgaben

- ☑ Operatorpräzedenz
- ☑ Kompilier- und Laufzeitfehler
- ❑ Short-circuiting
- ❑ Escapen von Strings
- ❑ Übung!

Aufgabe 1



```java
class MyClass{
    public static void main(String[] args){
        System.out.println(9 / 3 * 2 + 1 + "" + 2 * 3 + (4 * 3) % 3);
    }
}
```

"760"

Aufgabe 2

```java
class MyClass{
    public static void main(String[] args){
        System.out.println(8 / 5 + 0.5 + 5 / 2 + (8 % 3) * 1.0);
    }
}
```

5.5

Short-circuiting

- && (Logisches UND): Wenn der erste Ausdruck false ist, wird der zweite nicht ausgewertet.
  - *Beispiel:* false && 1/0 == Integer.MAX_VALUE
- || (Logisches ODER): Wenn der erste Ausdruck true ist, wird der zweite nicht ausgewertet.
  - *Beispiel:* true || 1/0 == Integer.MAX_VALUE

Checkliste - *println* Aufgaben

- ☑ Operatorpräzedenz
- ☑ Kompilier- und Laufzeitfehler
- ☑ Short-circuiting
- ❑ Escapen von Strings
- ❑ Übung!

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

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

Checkliste - `println` Aufgaben

- ☑ Operatorpräzedenz
- ☑ Kompilier- und Laufzeitfehler
- ☑ Short-circuiting
- ☑ Escapen von Strings
- ❑ Übung!

Aufgabe 3

```java
class MyClass{
    public static void main(String[] args){
        System.out.println(7 > 6 && (3 + ">" + 2) == "3 > 2" || 4 % 2 == 0 % 0);
    }
}
```

Exception (Division by Zero)

Aufgabe 4

```java
class MyClass{
    public static void main(String[] args){
        int a = 0;
        int b = 5;
        System.out.println((a > 0) && (b / a > 1)); // Was wird ausgegeben? Warum?
    }
}
```

False

Aufgabe 5

```java
class MyClass{
    public static void main(String[] args){
        int x = 3;
        int y = 10;
        System.out.println((x < y) || (x > 5) && (y++ == 10)); // Was wird ausgegeben? Was ist der Wert von y?
    }
}
```

True, y=10

Aufgabe 6

```java
class MyClass{
    public static void main(String[] args){
        int a = 3;
        int b = 4;
        System.out.println((a++ == 3) && (++b == 5)); // Was wird ausgegeben? Was sind die Werte von a und b nach dem Aufruf?
    }
}
```

True, a=4, b=5

Aufgabe 7

```java
class MyClass{
    public static void main(String[] args){
        int x = 2;
        int y = 8;
        System.out.println((x < y || ++x == 3) && (x++ == 3 || y-- > 5)); // Was wird ausgegeben?
    }
}
```

True, x=3, y=7

Aufgabe 8

```java
class MyClass{
    public static void main(String[] args){
        int m = 7;
        int n = 9;
        System.out.println((m == 7) || (n-- == 9) && (++m == 8) || (m == 9)); // Was wird ausgegeben?
    }
}
```

True, m=7, n=9

Aufgabe 1 (3)

Gegeben sei eine Methode `main` in einer Java Klasse.

```
public static void main(String[] args) {
    /* body */
}
```

Die folgenden Anweisungen sollen als "Body" (Rumpf) anstelle des Kommentars `/* body */` eingefügt werden. Geben Sie für jede Anweisung an, was für eine Ausgabe erzeugt wird – entweder was gedruckt wird, oder ob ein Laufzeitfehler auftritt (schreiben Sie "Exception"), oder ob der Compiler einen Fehler feststellt (schreiben Sie "Compile-Fehler"). Achten Sie auf die korrekte Formatierung der verschiedenen Typen, also z.B. 7.0 statt 7 für eine reelle Zahl (`double`).

1. `System.out.println(11 + 16 / 4 * 2 +  (4 + ">") + 4 * 2);`

   ---

2. `System.out.println(20 / 10 % 6 + 3 / 2 + (double) 5 / 4 + 3 / 4);`

   ---

3. `System.out.println( ( 11 % 4 ) >  2  && 9 > (16 / (2 / 4)) && 1 % 8 <  0);`

   ---

1. `System.out.println(11 + 16 / 4 * 2 +  (4 + ">") + 4 * 2);`

   194>8

1. `System.out.println(11 + 16 / 4 * 2 +  (4 + ">") + 4 * 2);`

   194>8

2. `System.out.println(20 / 10 % 6 + 3 / 2 + (double) 5 / 4 + 3 / 4);`

   4.25

1. `System.out.println(11 + 16 / 4 * 2 +  (4 + ">") + 4 * 2);`

   194>8 +1

2. `System.out.println(20 / 10 % 6 + 3 / 2 + (double) 5 / 4 + 3 / 4);`

   4.25 +1

3. `System.out.println( ( 11 % 4 ) >  2  && 9 > (16 / (2 / 4)) && 1 % 8 <  0);`

   Exception +1

Übung 2 - Vorbesprechung

Git, EBNF

Kahoot