

PSE – Vorkurs Tag 2

Tobias, Philipp, Linus, Tillmann

FIUS - Fachgruppe Informatik Universität Stuttgart

10. Juli 2025



am Anfang immer Vortages recap?	3
alles einheitlich mit minted block oder auch inline?	4

Recap Tag 1

am Anfang immer Vortages recap?



Boolean – Wahr oder Falsch

- ▶ Datentyp mit zwei Werten: `true` oder `false`
- ▶ Wahrheitswerte speichern und verarbeiten
- ▶ z.B: `heuteDienstag = true;`

```
1  boolean heuteDienstag = true;
```

alles einheitlich mit minted block oder auch inline?

Arithmetische Boolean Operatoren

Wenn a und b Zahlen sind, prüfen diese Operatoren Beziehungen zwischen ihnen:

► `a == b` Wahr, wenn a gleich b ist

```
1  boolean result = (a == b); // result true, wenn a gleich b
```

Arithmetische Boolean Operatoren

Wenn a und b Zahlen sind, prüfen diese Operatoren Beziehungen zwischen ihnen:

- ▶ `a == b` Wahr, wenn a gleich b ist

```
1  boolean result = (a == b); // result true, wenn a gleich b
```

- ▶ `a != b` Wahr, wenn a ungleich b ist

```
1  boolean result = (a != b); // result true, wenn a ungleich b
```

Arithmetische Boolean Operatoren

Wenn a und b Zahlen sind, prüfen diese Operatoren Beziehungen zwischen ihnen:

- ▶ `a == b` Wahr, wenn a gleich b ist

```
1  boolean result = (a == b); // result true, wenn a gleich b
```

- ▶ `a != b` Wahr, wenn a ungleich b ist

```
1  boolean result = (a != b); // result true, wenn a ungleich b
```

- ▶ `a < b` Wahr, wenn a kleiner als b ist

```
1  boolean result = (a < b); // result true, wenn a kleiner als b
```

Arithmetische Boolean Operatoren

Wenn a und b Zahlen sind, prüfen diese Operatoren Beziehungen zwischen ihnen:

- ▶ `a == b` Wahr, wenn a gleich b ist

```
1  boolean result = (a == b); // result true, wenn a gleich b
```

- ▶ `a != b` Wahr, wenn a ungleich b ist

```
1  boolean result = (a != b); // result true, wenn a ungleich b
```

- ▶ `a < b` Wahr, wenn a kleiner als b ist

```
1  boolean result = (a < b); // result true, wenn a kleiner als b
```

- ▶ analog bei `a > b`, `a <= b`, `a >= b`

Boolean Operatoren

► $\neg a \rightarrow \text{nicht } a$

Boolean Operatoren

► $!a \rightarrow$ nicht a

► $a \ \&\& \ b \rightarrow$ a UND b

Boolean Operatoren

► `!a` → nicht a

► `a && b` → a UND b

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

Boolean Operatoren

► `!a` → nicht a

► `a && b` → a UND b

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

► `a || b` → a ODER b

Boolean Operatoren

► `!a` → nicht a

► `a && b` → a UND b

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

► `a || b` → a ODER b

a	b	a b
true	true	true
true	false	true
false	true	true
false	false	false

Boolean Operatoren

► `!a` → nicht a

► `a && b` → a UND b

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

► `a || b` → a ODER b

a	b	a b
true	true	true
true	false	true
false	true	true
false	false	false

Klammern priorisieren:

► `true || false && false`

► `(true || false) && false`

Boolean Operatoren

► `!a` → nicht a

► `a && b` → a UND b

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

► `a || b` → a ODER b

a	b	a b
true	true	true
true	false	true
false	true	true
false	false	false

Klammern priorisieren:

► `true || false && false`
⇒ `true`

► `(true || false) && false`

Boolean Operatoren

► `!a` → nicht a

► `a && b` → a UND b

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

► `a || b` → a ODER b

a	b	a b
true	true	true
true	false	true
false	true	true
false	false	false

Klammern priorisieren:

► `true || false && false`
⇒ true

► `(true || false) && false`
⇒ false

if-Verzweigung

- ▶ Ausführung nur wenn Bedingung true
- ▶ Syntax:

```
1  if (Bedingung) {  
2    // Code bei true  
3  }
```

if-Verzweigung

- ▶ Ausführung nur wenn Bedingung true
- ▶ Syntax:

```
1  if (Bedingung) {  
2      // Code bei true  
3  }
```

- ▶ Beispiel:

```
1  boolean heuteDienstag = true;  
2  if (heuteDienstag) {  
3      System.out.println("Crazyyy heute ist Dinestag");  
4  }
```

Crazyyy heute ist Dinestag

if-Verzweigung

- ▶ geht genauso mit `int` etc.

if-Verzweigung

- ▶ geht genauso mit int etc.
- ▶ Beispiel:

```
1  int Lieblingszahl = 42;  
2  if (Lieblingszahl == 42) {  
3      System.out.println("Du bist ein Highperformer!");  
4  }
```

Du bist ein Highperformer!

if-else

- ▶ erweitert die if Anweisung
- ▶ "wenn if Bedingung nicht erfüllt dann mach folgendes..."
- ▶ Syntax:

```
1  if (Bedingung) {  
2      // Code bei Bedingung true  
3  } else {  
4      // Code bei Bedingung false  
5  }
```

if-else

- ▶ erweitert die if Anweisung
- ▶ "wenn if Bedingung nicht erfüllt dann mach folgendes..."
- ▶ Syntax:

```
1  if (Bedingung) {  
2    // Code bei Bedingung true  
3  } else {  
4    // Code bei Bedingung false  
5  }
```

- ▶ Beispiel:

```
1  boolean heuteDonnerstag = false;  
2  if (heuteDienstag) {  
3    System.out.println("endlich Wochenende");  
4  } else {  
5    System.out.println(":( bestimmt ist bald wieder Donnerstag");  
6  }
```

if-else

► Beispiel:

```
1  boolean heuteDonnerstag = false;
2  if (heuteDienstag) {
3      System.out.println("endlich Wochenende");
4  } else {
5      System.out.println(":( bestimmt ist bald wieder Donnerstag");
6  }
```

```
:( bestimmt ist bald wieder Donnerstag
```

else if

- ▶ else if prüft mehrere Bedingungen
- ▶ Beispiel:

```
1      if (note == 1) {  
2          System.out.println("Sehr gut");  
3      } else if (note == 2) {  
4          System.out.println("Gut");  
5      } else if (note == 3) {  
6          System.out.println("Befriedigend");  
7      } else {  
8          System.out.println("Ausreichend oder schlechter");  
9      }
```


while-Schleife

- ▶ Wiederholt Anweisungen, solange eine Bedingung true ist
- ▶ Syntax: `while (Bedingung) { /* Code */ }`

while-Schleife

- ▶ Wiederholt Anweisungen, solange eine Bedingung true ist
- ▶ Syntax: `while (Bedingung) { /* Code */ }`
- ▶ Beispiel:

```
1 Scanner scanner = new Scanner(System.in);  
2 String eingabe = "";  
3  
4 while (!eingabe.equals("ok")) {  
5     System.out.println("Bitte 'ok' eingeben:");  
6     eingabe = scanner.nextLine();  
7 }
```

Bitte 'ok' eingeben:

...

Bitte 'ok' eingeben:

(Benutzer tippt ök) → Schleife endet

break

- ▶ Mit break kann man eine Schleife vorzeitig beenden

```
1 Scanner scanner = new Scanner(System.in);
2
3 String passwort = "diegrillung";
4 String abbruchBedingung = "abbruch";
5 String momentaneEingabe = "";
6
7 while (!momentaneEingabe.equals(passwort)) {
8     momentaneEingabe = scanner.nextLine();
9
10    if (momentaneEingabe.equals(abbruchBedingung)) {
11        break;
12    }
13 }
14 System.out.println("I'm in");
```

WURST TOGETHER



Was ist eine for-Schleife?

- ▶ Wiederholt Anweisungen eine festgelegte Anzahl von Malen
- ▶ Syntax:

```
1  for (Start; Bedingung; Schritt) {  
2      // Schleifenrumpf  
3  }
```

Somit wird aus...

```
1      int momentaneWurst = 4
2      int letzteWurst = 8
3      while (momentaneWurst <= letzteWurst) {
4          System.out.println("Schmeiß Wurst Nr." + momentaneWurst + "auf den Grill");
5          momentaneWurst++;
6      }
```

ganz simpel...

```
1      for (int i = 4; i <= 8; i++) {
2          System.out.println("Schmeiß Wurst Nr." + i + "auf den Grill");
3      }
```