PSE – Vorkurs Tag 2

Tobias, Philipp, Linus, Tillmann

FIUS - Fachgruppe Informatik Universität Stuttgart

7. Juli 2025

Recap Tag 1

am Anfang immer Vortages recap?

Was sind Verzweigungen?

- Programme müssen Entscheidungen treffen

  item oder bullet item?

- Beispiel: Links oder rechts gehen?

Boolean – Wahr oder Falsch

- Datentyp mit zwei Werten: `true` und `false`
- Wird für Bedingungen verwendet
- z.B: `heuteDienstag = true;`

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

Boolean – Wahr oder Falsch

- Datentyp mit zwei Werten: `true` und `false`
- Wird für Bedingungen verwendet
- z.B: `heuteDienstag = true;`

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

- Wird oft in Bedingungen verwendet, z.B. in if-Anweis

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

- `!a` $\rightarrow$ nicht a

Boolean Operatoren

- `!a` $\rightarrow$ nicht a
- `a && b` $\rightarrow$ a UND b

Boolean Operatoren

- `!a` $\rightarrow$ nicht a
- `a && b` $\rightarrow$ a UND b
- `a || b` $\rightarrow$ a ODER b

Boolean Operatoren

- `!a` $\rightarrow$ nicht a
- `a && b` $\rightarrow$ a UND b
- `a || b` $\rightarrow$ a ODER b

Klammern priorisieren:

- `true || false && false` $\rightarrow$

Boolean Operatoren

- `!a` $\rightarrow$ nicht a
- `a && b` $\rightarrow$ a UND b
- `a || b` $\rightarrow$ a ODER b

Klammern priorisieren:

- `true || false && false` $\rightarrow$ `true`

Boolean Operatoren

- `!a` $\rightarrow$ nicht a
- `a && b` $\rightarrow$ a UND b
- `a || b` $\rightarrow$ a ODER b

Klammern priorisieren:

- `true || false && false` $\rightarrow$ `true`
- `(true || false) && false` $\rightarrow$

Boolean Operatoren

- `!a` $\rightarrow$ nicht a
- `a && b` $\rightarrow$ a UND b
- `a || b` $\rightarrow$ a ODER b

Klammern priorisieren:

- `true || false && false` $\rightarrow$ `true`
- `(true || false) && false` $\rightarrow$ `false`

if-Verzweigung

- Boolescher Ausdruck entscheidet
- Ausführung nur wenn `true`
- Syntax:

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

if-Verzweigung

- Boolescher Ausdruck entscheidet
- Ausführung nur wenn `true`
- Syntax:

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

- Beispiel:

```
1  boolean a = true;
2  if (a) {
3    System.out.println("a ist wahr");
4  }
```

```
a ist wahr
```

- Erweiterung der if-Anweisung
- Ausführung bei `true` oder `false`
- Syntax:

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

- Erweiterung der if-Anweisung
- Ausführung bei `true` oder `false`
- Syntax:

```
if (Bedingung) {
  // Code bei Bedingnung true
} else {
  // Code bei Bedingung false
}
```

- Beispiel:

```
boolean a = false;
if (a) {
  System.out.println("a ist wahr");
} else {
  System.out.println("a ist falsch");
}
```

if-else-Verzweigung

- Beispiel:

```
1  boolean a = false;
2  if (a) {
3    System.out.println("a ist wahr");
4  } else {
5    System.out.println("a ist falsch");
6  }
```

```
a ist falsch
```

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:

```java
Scanner scanner = new Scanner(System.in);
String eingabe = "";

while (!eingabe.equals("ok")) {
  System.out.println("Bitte 'ok' eingeben:");
  eingabe = scanner.nextLine();
}
```

```
Bitte ’ok’ eingeben:
...
Bitte ’ok’ eingeben:
(Benutzer tippt ök") → Schleife endet
```

break

- Mit `break` kann man eine Schleife vorzeitig beenden.
- Dies ist nützlich, wenn eine Abbruchbedingung innerhalb der Schleife erkannt wird.

```
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?

- Eine `for`-Schleife hilft dir, etwas **mehrmals** zu tun.
- Syntax:

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

- Ablauf:
  1. Wo starten wir? (z.B. bei 0)
  2. Bedingung noch erfüllt?
  3. Falls ja, Code im Block ausführen
  4. Zähler ändern i.d.R. einen hochzählen

bessere Erklärung

Somit wird aus...

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

ganz simpel...

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