

# PSE – Vorkurs Tag 2

Tobias, Philipp, Linus, Tillmann

FIUS - Fachgruppe Informatik Universität Stuttgart

6. Juli 2025



am Anfang immer Vortages recap? . . . . .	3
item oder bullet item? . . . . .	4
mint style festlegen (z.B. friendly) . . . . .	5

# 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;`

mint style festlegen (z.B. friendly)

# 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

►  $\neg a \rightarrow$  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` → nicht a
- ▶ `a && b` → a UND b
- ▶ `a || b` → a ODER b

Klammern priorisieren:

- ▶ `true || false && false` →

# Boolean Operatoren

- ▶ `!a` → nicht a
- ▶ `a && b` → a UND b
- ▶ `a || b` → a ODER b

Klammern priorisieren:

- ▶ `true || false && false` → `true`

# Boolean Operatoren

- ▶ `!a` → nicht a
- ▶ `a && b` → a UND b
- ▶ `a || b` → a ODER b

Klammern priorisieren:

- ▶ `true || false && false` → `true`
- ▶ `(true || false) && false` →

# Boolean Operatoren

- ▶ `!a` → nicht a
- ▶ `a && b` → a UND b
- ▶ `a || b` → a ODER b

Klammern priorisieren:

- ▶ `true || false && false` → `true`
- ▶ `(true || false) && false` → `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  }
```

# 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  }  
  
//Ausgabe: a ist wahr
```

# if-else-Verzweigung

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

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

# if-else-Verzweigung

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

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

- ▶ 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  }
```

# if-else-Verzweigung

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

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

- ▶ 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  }  
  
//Ausgabe: 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:

```
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 }
8
9 // Ausgabe
10 // Bitte 'ok' eingeben:
11 // ...
12 // Bitte 'ok' eingeben:
13 // (Benutzer tippt "ok") → Schleife endet
```



# while-Schleife mit break

- break beendet eine Schleife sofort

```
1 Scanner scanner = new Scanner(System.in);
2
3 String passwort = "diegrillung";
4 String abbruch = "abbruch";
5 String eingabe = "";
6
7 while (true) {
8     System.out.println("Bitte Passwort eingeben:");
9     eingabe = scanner.nextLine();
10
11     if (eingabe.equals(passwort)) {
12         System.out.println("Zugriff erlaubt");
13         break;
14     }
15
16     if (eingabe.equals(abbruch)) {
17         System.out.println("Abbruch durch Benutzer");
18         break;
19     }
20 }
21 System.out.println("Programm beendet");
```

