PSE – Vorkurs Tag 3

Linus, Philipp, Tillmann, Tobias

FIUS - Fachgruppe Informatik Universität Stuttgart

08.09.2025

Recap Tag 2 - Bedingungen & Booleans

Recap Tag 2 - Bedingungen & Booleans

- Boolean: Wahrheitswerte `true` und `false`

Recap Tag 2 - Bedingungen & Booleans

- Boolean: Wahrheitswerte `true` und `false`
- Vergleiche: ==, !=, <, >, <=, >=

Recap Tag 2 - Bedingungen & Booleans

- Boolean: Wahrheitswerte `true` und `false`
- Vergleiche: ==, !=, <, >, <=, >=
- Logische Operatoren: && (UND), || (ODER), ! (NICHT)

Recap Tag 2 - Bedingungen & Booleans

- Boolean: Wahrheitswerte `true` und `false`
- Vergleiche: ==, !=, <, >, <=, >=
- Logische Operatoren: && (UND), || (ODER), ! (NICHT)
- `if`, `else`, `else if` zur Entscheidungslogik

```
1  if (x > 5 && x < 10) {
2    System.out.println("x ist zwischen 5 und 10");
3  } else {
4    System.out.println("x ist außerhalb");
5  }
```

Recap Tag 2 - Schleifen

Recap Tag 2 - Schleifen

- `while`-Schleife: läuft, solange Bedingung `true` ist

```
1  while (!eingabe.equals("ok")) {
2    eingabe = scanner.nextLine();
3  }
```

Recap Tag 2 - Schleifen

- `while`-Schleife: läuft, solange Bedingung `true` ist

```
1  while (!eingabe.equals("ok")) {
2    eingabe = scanner.nextLine();
3  }
```

- `for`-Schleife: Zählergesteuerte Schleife

```
1  for (int i = 0; i < 5; i++) {
2    System.out.println(i);
3  }
```

Recap Tag 2 - Schleifen

- `while`-Schleife: läuft, solange Bedingung `true` ist

```
while (!eingabe.equals("ok")) {
  eingabe = scanner.nextLine();
}
```

- `for`-Schleife: Zählergesteuerte Schleife

```
for (int i = 0; i < 5; i++) {
  System.out.println(i);
}
```

- `break`: beendet Schleife vorzeitig

Angenommen . . .

Angenommen . . .

- Notendurchschnitt von zwei Studis berechnen

```java
// Melanie
double m1 = 1.7, m2 = 2.3, m3 = 1.3;
double melanieDurchschnitt = (m1 + m2 + m3) / 3;
System.out.println("Melanies Schnitt: " + melanieDurchschnitt);

// Paul
double p1 = 4.0, p2 = 2.3, p3 = 3.3;
double paulDurchschnitt = (p1 + p2 + p3) / 3;
System.out.println("Pauls Schnitt: " + paulDurchschnitt);
```

```
Melanies Schnitt: 1.7666666666666666
Pauls Schnitt: 3.1999999999999997
```

Warum so nicht?!

Warum so nicht?!

- **Redundanz:** Gleicher Code mehrfach

Warum so nicht?!

- **Redundanz:** Gleicher Code mehrfach
- **Fehleranfällig:** Zahlendreher möglich

Warum so nicht?!

- **Redundanz:** Gleicher Code mehrfach
- **Fehleranfällig:** Zahlendreher möglich
- **Schlecht wartbar:** Änderungen mehrfach nötig

Warum so nicht?!

- **Redundanz:** Gleicher Code mehrfach
- **Fehleranfällig:** Zahlendreher möglich
- **Schlecht wartbar:** Änderungen mehrfach nötig
- **Nicht wiederverwendbar:** Nur an dieser Stelle nutzbar

Warum so nicht?!

- **Redundanz:** Gleicher Code mehrfach
- **Fehleranfällig:** Zahlendreher möglich
- **Schlecht wartbar:** Änderungen mehrfach nötig
- **Nicht wiederverwendbar:** Nur an dieser Stelle nutzbar
- **Unübersichtlich:** Logik geht unter

Warum so nicht?!

- **Redundanz:** Gleicher Code mehrfach
- **Fehleranfällig:** Zahlendreher möglich
- **Schlecht wartbar:** Änderungen mehrfach nötig
- **Nicht wiederverwendbar:** Nur an dieser Stelle nutzbar
- **Unübersichtlich:** Logik geht unter

$\longrightarrow$ Funktionen ermöglichen Wiederverwendung von Code!

Was sind eigentlich Funktionen?

Was sind eigentlich Funktionen?

- Möglichkeit Code auszulagern

Was sind eigentlich Funktionen?

- Möglichkeit Code auszulagern
- Syntax:

```
1  Rückgabedatentyp Funktionsbezeichner(Datentyp Parametername,...){
2          ...
3          return Rückgabewert;
4   }
```

Was sind eigentlich Funktionen?

- Möglichkeit Code auszulagern
- Syntax:

```
1  Rückgabedatentyp Funktionsbezeichner(Datentyp Parametername,...){
2          ...
3          return Rückgabewert;
4   }
```

- in unserem Fall:

```
1  static void avg(String name, double note1, double note2, double note3) {
2      double avg = (note1 + note2 + note3) / 3;
3      System.out.println(name + "s Schnitt: " + avg);}
```

Schlüsselwort `static` ist hier notwendig wird in PSE genauer erklärt

return

return

- Funktionen können Werte zurückgeben

return

- Funktionen können Werte zurückgeben
- z.B. eine Zahl:

```
1  static int add(int num1, int num2){
2      int sum = num1 + num2;
3      return sum;
4  }
```

return

- Funktionen können Werte zurückgeben
- z.B. eine Zahl:


```
1  static int add(int num1, int num2){
2      int sum = num1 + num2;
3      return sum;
4  }
```

- Datentyp des `return`-Werts muss im Funktionskopf stehen

return

- Funktionen können Werte zurückgeben
- z.B. eine Zahl:


```
1  static int add(int num1, int num2){
2      int sum = num1 + num2;
3      return sum;
4  }
```

- Datentyp des `return`-Werts muss im Funktionskopf stehen
- Wenn kein Rückgabewert → `void` (wie vorhin)

Verwendung von `return`

Verwendung von `return`

- Rückgabewert wird zurück an den Aufrufer gegeben

Verwendung von `return`

- Rückgabewert wird zurück an den Aufrufer gegeben
- Speicherung in Variable

```
1  int result = add(3, 4);
2  System.out.println(result);
```

```
7
```

Verwendung von `return`

- Rückgabewert wird zurück an den Aufrufer gegeben
- Speicherung in Variable

```java
int result = add(3, 4);
System.out.println(result);
```

```
7
```

- `return` beendet Funktion sofort

Verwendung von `return`

- Rückgabewert wird zurück an den Aufrufer gegeben
- Speicherung in Variable

```java
int result = add(3, 4);
System.out.println(result);
```

```
7
```

- `return` beendet Funktion sofort
- Code nach `return` wird nicht mehr ausgeführt

Code together: Promillerechner

Aufgabe: Schreibe eine Funktion `berechnePromille`, die anhand der Getränkemenge (in Litern), des Alkoholgehalts (Vol-%) und des Körpergewichts die Blutalkoholkonzentration berechnet.

Formeln:

$$\text{Promille} \approx \frac{\text{Alkohol in Gramm}}{\text{Körpergewicht in kg} \times 0{,}65}$$

$$\text{Alkohol in Gramm} = \text{Getränkemenge in Liter} \times \text{Vol\%} \times 8$$

Die `main`-Funktion

Die `main`-Funktion

- Einstiegspunkt jedes Java-Programms

Die `main`-Funktion

- Einstiegspunkt jedes Java-Programms
- Wird beim Start automatisch aufgerufen

Die `main`-Funktion

- Einstiegspunkt jedes Java-Programms
- Wird beim Start automatisch aufgerufen
- Muss exakt so deklariert sein:

```
1  public static void main(String[] args) {
2      // Hier beginnt das Programm
3  }
```

Sichtbarkeit von Variablen (Scopes)

Sichtbarkeit von Variablen (Scopes)

- Variablen sind sichtbar im aktuellen und in allen tieferen Blöcken

```
1  public static void main(String[] args) {
2      int y = 10;
3
4      if (y > 5) {
5          System.out.println(y); // OK - Zugriff auf äußere Variable
6      }
7  }
```

Sichtbarkeit von Variablen (Scopes)

Sichtbarkeit von Variablen (Scopes)

- In äußeren Blöcken sind sie nicht sichtbar

```java
public static void main(String[] args) {
    int x = 5;
}
public static void andereFunktion() {
    System.out.println(x); // Fehler! x ist hier nicht sichtbar
}
```

Sichtbarkeit von Variablen (Scopes)

- In äußeren Blöcken sind sie nicht sichtbar

```java
public static void main(String[] args) {
    int x = 5;
}
public static void andereFunktion() {
    System.out.println(x); // Fehler! x ist hier nicht sichtbar
}
```

- Gilt für alle Blöcke: Funktionen, if, Schleifen, etc.

Check Yourself!

Frage 1: Was gibt die void-Deklaration bei einer Funktion an?

```
Die Funktion gibt keinen Wert zurück
```

Frage 2: Welche Aussage ist richtig?

```
1  public static int multiply(int a, int b) {
2      int result = a * b;
3      return result;
4  }
```

Es gibt keinen Fehler im Code

Frage 3: Bilde die erste Zeile einer Funktion, die eine Note als Parameter erhält und nichts zurückgibt?

```
1  public static void printGrade(double grade) {
2      // für Lösung nur Zeile 1 relevant
3  }
```

Frage 4: Welche Aussage über Funktionen in Java ist richtig??

```
Funktionen in Java können andere Funktionen aufrufen
```

Frage 5: Was wird ausgegeben?

```java
if (true) {
    int y = 8;
}
System.out.println(y);
```

Fehler

Frage 6: Was wird ausgegeben?

```
1  int i = 0;
2  for (i = 0; i < 3; i++) {
3      System.out.print(i);
4  }
5  System.out.print(i);
```

```
0123
```

Was würde ausgegeben werden, wenn Zeile 1 entfernt wird?

Frage 7: Was wird ausgegeben?

```java
public static void main(String[] args) {
    int a = 5;
    zeigeA(a);
    System.out.print(a);
}

public static void zeigeA(int a) {
    a = a + 1;
    System.out.print(a);
}
```

65

Was ist ein Array?

Was ist ein Array?

- Arrays speichern mehrere Werte des selben Datentyps

Was ist ein Array?

- Arrays speichern mehrere Werte des selben Datentyps
- Länge ist beim Erstellen fest und unveränderlich

Was ist ein Array?

- Arrays speichern mehrere Werte des selben Datentyps
- Länge ist beim Erstellen fest und unveränderlich
- Elemente sind geordnet und über einen Index erreichbar

Was ist ein Array?

- Arrays speichern mehrere Werte des selben Datentyps
- Länge ist beim Erstellen fest und unveränderlich
- Elemente sind geordnet und über einen Index erreichbar
- Syntax:

```
1  Datentyp[] Bezeichner = new Datentyp[ArrayLänge];
```

Was ist ein Array?

- Arrays speichern mehrere Werte des selben Datentyps
- Länge ist beim Erstellen fest und unveränderlich
- Elemente sind geordnet und über einen Index erreichbar
- Syntax:

```
1  Datentyp[] Bezeichner = new Datentyp[ArrayLänge];
```

- z.B.: leeres Array für 5 Ganzzahlen

```
1  int[] numbers = new int[5];
```

Zugriff auf Array-Elemente

Zugriff auf Array-Elemente

- Jedes Element hat einen Index

Zugriff auf Array-Elemente

- Jedes Element hat einen Index
- über Index lässt sich auf Elemente zugreifen

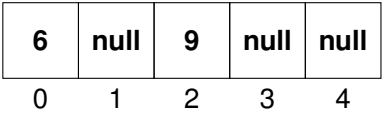
```
1  int[] numbers = new int[5];
2  numbers[0] = 6;
3  numbers[2] = 9;
```

6	null	9	null	null
0	1	2	3	4

Zugriff auf Array-Elemente

- Jedes Element hat einen Index
- über Index lässt sich auf Elemente zugreifen

```
1  int[] numbers = new int[5];
2  numbers[0] = 6;
3  numbers[2] = 9;
```



- Arrays können auch direkt mit Werten initialisiert werden

```
1  String[] grillSachen = {"würstchen", "steak",
2                          "grillkäse", "maiskolben"};
```

Arrays durchlaufen mit `.length`

Arrays durchlaufen mit `.length`

- speichert Länge des Arrays

```
1  Bezeichner.length
```

Arrays durchlaufen mit `.length`

- speichert Länge des Arrays

```
1  Bezeichner.length
```

- damit lässt sich über alle Elemente iterieren

```
1  System.out.println(grillSachen[1]);
2  for (int i = 0; i < grillSachen.length; i++) {
3      System.out.println(grillSachen[i]);
4  }
```

```
steak
würstchen
steak
grillkäse
maiskolben
```

Ende

- Folien und Aufgaben:
  `https://fius.de/index.php/pse-vk-folien/`

Wenn ihr Fragen habt, sagt Bescheid!