PSE - Vorkurs Tag 1

Linus, Philipp, Tillmann, Tobias

FIUS - Fachgruppe Informatik Universität Stuttgart

06.09.2025

To-Do: Entwicklungsumgebung einrichten

- Verbinde dich mit dem WLAN
- Installiere IntelliJ IDEA (Community Edition)
  ⚠️ Achtung: Ihr könnt auch eine andere IDE verwenden
- Installiere das JDK (Java Development Kit)
- Projekt erstellen

Bei Fragen einfach melden!

Danach geht's gleich weiter...

Erste Java Klasse

Nun schreiben wir folgenen Code in die `Main`-Klasse

```java
public class Main {
  public static void main(String[] args) {
    System.out.println("Hallo Welt!");
  }
}
```

⚠ Achtung: Was das alles bedeutet, werden wir euch dann in den nächsten Tagen erklären!

Erste Java Klasse

- Das Programm soll nun von euch gestartet werden
- Dazu klickt auf das grüne Dreieck oben rechts in der IDE
- Wenn alles korrekt eingerichtet ist, sollte die Ausgabe "Hallo Welt!" erscheinen. ⚠️
  Achtung: Wenn eine Fehlermeldung erscheint, überprüft bitte den Code alternativ fragt einen Tutor.

GitHub

GitHub ist eine Plattform, auf der man Code online speichern, verwalten und mit Anderen teilen kann. Es basiert auf Git, einem Versionskontrollsystem.

Warum GitHub für Java-Projekte?

Warum GitHub für Java-Projekte?

- **Sicherung:** Java-Code (Klassen, Methoden, Projekte) kann sicher gespeichert werden.

Warum GitHub für Java-Projekte?

- **Sicherung:** Java-Code (Klassen, Methoden, Projekte) kann sicher gespeichert werden.
- **Versionskontrolle:** Änderungen zwischen Versionen sind nachvollziehbar – praktisch beim Lernen oder bei Fehlern.

Warum GitHub für Java-Projekte?

- **Sicherung:** Java-Code (Klassen, Methoden, Projekte) kann sicher gespeichert werden.
- **Versionskontrolle:** Änderungen zwischen Versionen sind nachvollziehbar – praktisch beim Lernen oder bei Fehlern.
- **Zusammenarbeit:** Gemeinsames Arbeiten an Projekten und Feedback sind möglich.

Warum GitHub für Java-Projekte?

- **Sicherung:** Java-Code (Klassen, Methoden, Projekte) kann sicher gespeichert werden.
- **Versionskontrolle:** Änderungen zwischen Versionen sind nachvollziehbar – praktisch beim Lernen oder bei Fehlern.
- **Zusammenarbeit:** Gemeinsames Arbeiten an Projekten und Feedback sind möglich.
- **Überall verfügbar:** Zugriff von verschiedenen Orten – beispielsweise an der Uni, zuhause oder unterwegs.

Variablen

Variablen

- Variablen werden benötigt, um Werte im Programm zwischenzuspeichern

Variablen

- Variablen werden benötigt, um Werte im Programm zwischenzuspeichern
- Sie haben einen Namen (Bezeichner), einen Datentyp und speichern einen Wert

Variablen

- Variablen werden benötigt, um Werte im Programm zwischenzuspeichern
- Sie haben einen Namen (Bezeichner), einen Datentyp und speichern einen Wert
- Über den Namen kann später auf den gespeicherten Wert zugegriffen werden

Variablen

- Variablen werden benötigt, um Werte im Programm zwischenzuspeichern
- Sie haben einen Namen (Bezeichner), einen Datentyp und speichern einen Wert
- Über den Namen kann später auf den gespeicherten Wert zugegriffen werden

```
1  int alter = 20;
```

- `alter` ist der Name der Variable, `int` der Datentyp (ganzzahlige Zahl) und `20` der gespeicherte Wert

Konzept `null`

Konzept `null`

- `null` steht für „kein Wert“

```
1  String name = null; // name zeigt auf nichts
```

Konzept `null`

- `null` steht für „kein Wert“

```
1  String name = null; // name zeigt auf nichts
```

- Zugriff auf Methoden oder Eigenschaften einer `null`-Referenz führt zu einem Fehler "`NullPointerException`".

Konzept `null`

- `null` steht für „kein Wert“

```
1  String name = null; // name zeigt auf nichts
```

- Zugriff auf Methoden oder Eigenschaften einer `null`-Referenz führt zu einem Fehler "`NullPointerException`".

⚠ Achtung: `null` ist nicht dasselbe wie eine leere Zeichenkette `''` oder die Zahl 0.

Variablen

Variablen

- Deklaration $\rightarrow$ Variable wird erstellt **OHNE** einen Wert zuzuweisen

```
1  int zahl;
```

Variablen

- Deklaration → Variable wird erstellt **OHNE** einen Wert zuzuweisen

```
1  int zahl;
```

- Initialisierung → Variable wird sein **ERSTER** Wert zugewiesen

```
1  zahl = 42;
```

Variablen

- Deklaration → Variable wird erstellt **OHNE** einen Wert zuzuweisen

```
1  int zahl;
```

- Initialisierung → Variable wird sein **ERSTER** Wert zugewiesen

```
1  zahl = 42;
```

```
1  //In der Regel Deklaration und Initialisierung gleichzeitig
2  int zahl = 42;
3
4  //aber auch so möglich
5  int zahl;
6  // ...
7  zahl = 42;
```

Variablen

Variablen

- Variablen können auch später im Programm verändert werden

```
1  int alter = 20;
2  alter = 21; // alter ist jetzt 21
```

Datentypen in Java

Datentypen in Java

Typ	Beschreibung
`int`	Ganzzahlen (z. B. 1, 2, 3)
`float`	Kommazahlen (weniger genau)
`double`	Kommazahlen (hohe Genauigkeit)
`char`	Einzelnes Zeichen (z. B. 'a')
`short`	Kleine Ganzzahlen
`long`	Große Ganzzahlen
`boolean`	Wahrheitswert (`true` / `false`)
`String`	Text (z. B. "Hallo")

Operatoren in Java

Operatoren in Java

Operator	Bedeutung	Gültige Typen
+	Addition / Verkettung	Zahlen, String
-	Subtraktion	Zahlen
*	Multiplikation	Zahlen
/	Division	Zahlen
%	Modulo (Rest)	Ganzzahlen

Beispiel: `String + Zahl`

Beispiel: String + Zahl

```
1  String text = "Ergebnis: ";
2  int zahl = 5;
3  System.out.println(text + zahl);
```

Beispiel: String + Zahl

```
1  String text = "Ergebnis: ";
2  int zahl = 5;
3  System.out.println(text + zahl);
```

```
Ergebnis: 5
```

Kommentare in Java

Kommentare in Java

- `// Einzeilige Kommentare`
- `/* Mehrzeilige Kommentare */`

Kommentare in Java

- `// Einzeilige Kommentare`
- `/* Mehrzeilige Kommentare */`

```
1  float gewicht = 23.4; // Gewicht in kg
2
3  /*
4  Berechnet die Summe
5  zweier Zahlen
6  */
7  int summe = x + y;
```

Textausgabe in Java

Textausgabe in Java

Problem: Wir wollen irgendwas in der Konsole ausgeben

Textausgabe in Java

Problem: Wir wollen irgendwas in der Konsole ausgeben

```
1  System.out.print("Das ist eine Konsolenausgabe");
```

```
Das ist eine Konsolenausgabe
```

Textausgabe in Java

Problem: Wir wollen irgendwas in der Konsole ausgeben

```java
System.out.print("Das ist eine Konsolenausgabe");
```

```
Das ist eine Konsolenausgabe
```

```java
System.out.print("Das ist eine Konsolenausgabe");
System.out.print("Das ist noch eine Ausgabe");
```

Textausgabe in Java

Problem: Wir wollen irgendwas in der Konsole ausgeben

```
1  System.out.print("Das ist eine Konsolenausgabe");
```

```
Das ist eine Konsolenausgabe
```

```
1  System.out.print("Das ist eine Konsolenausgabe");
2  System.out.print("Das ist noch eine Ausgabe");
```

```
Das ist eine KonsolenausgabeDas ist noch eine Ausgabe
```

Zeilenumbruch: `println` und `\n`

Zeilenumbruch: `println` und `\n`

```
1  System.out.println("1. Ausgabe");
2  System.out.println("2. Ausgabe");
```

Zeilenumbruch: println und \n

```
1  System.out.println("1. Ausgabe");
2  System.out.println("2. Ausgabe");
```

```
1. Ausgabe
2. Ausgabe
```

Zeilenumbruch: `println` und `\n`

```
1  System.out.println("1. Ausgabe");
2  System.out.println("2. Ausgabe");
```

```
1. Ausgabe
2. Ausgabe
```

```
1  System.out.print("1. Ausgabe\n");
2  System.out.print("2. Ausgabe");
```

Zeilenumbruch: `println` und `\n`

```
1  System.out.println("1. Ausgabe");
2  System.out.println("2. Ausgabe");
```

```
1. Ausgabe
2. Ausgabe
```

```
1  System.out.print("1. Ausgabe\n");
2  System.out.print("2. Ausgabe");
```

```
1. Ausgabe
2. Ausgabe
```

Ausgabe von Variablen

Ausgabe von Variablen

Man kann nicht nur Text, sondern auch Variablen ausgeben:

Ausgabe von Variablen

Man kann nicht nur Text, sondern auch Variablen ausgeben:

```
1  int number = 42;
2  System.out.println(number);
```

```
42
```

Ausgabe von Variablen

Man kann nicht nur Text, sondern auch Variablen ausgeben:

```java
int number = 42;
System.out.println(number);
```

```
42
```

Oder beides kombiniert:

```java
int number = 42;
System.out.println("Meine Lieblingszahl ist " + number + "!");
```

Ausgabe von Variablen

Man kann nicht nur Text, sondern auch Variablen ausgeben:

```java
int number = 42;
System.out.println(number);
```

```
42
```

Oder beides kombiniert:

```java
int number = 42;
System.out.println("Meine Lieblingszahl ist " + number + "!");
```

```
Meine Lieblingszahl ist 42!
```

Check Yourself!

Frage 1: Was wird ausgegeben?

```
1  System.out.println(4 + 5);
```

Frage 1: Was wird ausgegeben?

```java
System.out.println(4 + 5);
```

```
9
```

Frage 2: Was wird ausgegeben?

```
1  System.out.println(3 + "4");
```

Frage 2: Was wird ausgegeben?

```
1  System.out.println(3 + "4");
```

```
34
```

Frage 3: Was wird ausgegeben?

```
1  System.out.println(2);
2  System.out.println(3);
```

Frage 3: Was wird ausgegeben?

```
1  System.out.println(2);
2  System.out.println(3);
```

```
23
```

Frage 4:Was wird ausgegeben?

```
1  System.out.println(13 % 3);
```

Frage 4:Was wird ausgegeben?

```java
System.out.println(13 % 3);
```

```
1
```

Frage 5:Was wird ausgegeben?

```
1  System.out.println(10 + ((3 * 4)+ "7"));
```

Frage 5:Was wird ausgegeben?

```
1  System.out.println(10 + ((3 * 4)+ "7"));
```

```
10127
```

Frage 6: Was wird ausgegeben?

```
1  int a = 3;
2  // a = 3 + 4;
3  System.out.println(a);
```

Frage 6: Was wird ausgegeben?

```java
int a = 3;
// a = 3 + 4;
System.out.println(a);
```

```
3
```

Frage 7:Was wird ausgegeben?

```
1  System.out.println(10 + "6" * 4);
```

Frage 7:Was wird ausgegeben?

```
1  System.out.println(10 + "6" * 4);
```

```
Fehler
```

Frage 8:Was wird ausgegeben?

```
1  int a = 8 * 4;
2  a = a / 5;
3  System.out.println(a);
```

Frage 8:Was wird ausgegeben?

```
1  int a = 8 * 4;
2  a = a / 5;
3  System.out.println(a);
```

```
6
```

Frage 9: Was wird ausgegeben?

```
1  float a = 8 * 4;
2  a = a / 5;
3  System.out.println(a + "3");
```

Frage 9: Was wird ausgegeben?

```
1  float a = 8 * 4;
2  a = a / 5;
3  System.out.println(a + "3");
```

```
6.43
```

Eingabe mit `Scanner`

Eingabe mit `Scanner`

Benutzereingaben werden in Java mit der `Scanner` Klasse verarbeitet

Eingabe mit `Scanner`

Benutzereingaben werden in Java mit der `Scanner` Klasse verarbeitet

- Mit `Scanner` kann man Benutzereingaben aus der Konsole einlesen.

Eingabe mit `Scanner`

Benutzereingaben werden in Java mit der `Scanner` Klasse verarbeitet

- Mit `Scanner` kann man Benutzereingaben aus der Konsole einlesen.
- Dafür muss die Klasse `Scanner` aus dem Paket `java.util` importiert werden.

Eingabe mit `Scanner`

Benutzereingaben werden in Java mit der `Scanner` Klasse verarbeitet

- Mit `Scanner` kann man Benutzereingaben aus der Konsole einlesen.
- Dafür muss die Klasse `Scanner` aus dem Paket `java.util` importiert werden.
- Imports steht ganz oben im Java-Quelltext

Eingabe mit `Scanner`

Benutzereingaben werden in Java mit der `Scanner` Klasse verarbeitet

- Mit `Scanner` kann man Benutzereingaben aus der Konsole einlesen.
- Dafür muss die Klasse `Scanner` aus dem Paket `java.util` importiert werden.
- Imports steht ganz oben im Java-Quelltext

```
1  import java.util.Scanner;
```

Eingabe mit `Scanner`

Benutzereingaben werden in Java mit der `Scanner` Klasse verarbeitet

- Mit `Scanner` kann man Benutzereingaben aus der Konsole einlesen.
- Dafür muss die Klasse `Scanner` aus dem Paket `java.util` importiert werden.
- Imports steht ganz oben im Java-Quelltext

```java
import java.util.Scanner;
```

```java
Scanner scanner = new Scanner(System.in);
String name = scanner.nextLine();
System.out.println(name);
```

Syntaxfehler - Beispiele

Syntaxfehler - Beispiele

- **Syntaxfehler:** Code verletzt Java-Regeln (z. B. fehlendes ‘;‘)

```
1  // Fehlendes Semikolon
2  System.out.println("Hallo")
3
4  // Variable nicht deklariert
5  zahl = 5;
```

Laufzeitfehler - Beispiele

- **Laufzeitfehler:** Fehler während der Ausführung (z. B. Division durch 0)

```
1  String text = null;
2  text.length(); // NullPointerException
3
4  int zahl = 5 / 0; // ArithmeticException
```

Fehler suche IDE

Fehler suche IDE

- IntelliJ IDEA bietet viele Tools zur Fehlersuche

Fehler suche IDE

- IntelliJ IDEA bietet viele Tools zur Fehlersuche
- Fehler werden rot unterstrichen und in der rechten Leiste angezeigt

Fehler suche IDE

- IntelliJ IDEA bietet viele Tools zur Fehlersuche
- Fehler werden rot unterstrichen und in der rechten Leiste angezeigt
- Wenn ihr über den Fehler hovert, seht ihr eine Beschreibung des Problems

Fehler suche IDE

- IntelliJ IDEA bietet viele Tools zur Fehlersuche
- Fehler werden rot unterstrichen und in der rechten Leiste angezeigt
- Wenn ihr über den Fehler hovert, seht ihr eine Beschreibung des Problems
- Nutze die integrierte Konsole, um Ausgaben und Fehler zu sehen

```
1  System.out.println("Lecker Grillerei!")
```

Fehler suche IDE

- IntelliJ IDEA bietet viele Tools zur Fehlersuche
- Fehler werden rot unterstrichen und in der rechten Leiste angezeigt
- Wenn ihr über den Fehler hovert, seht ihr eine Beschreibung des Problems
- Nutze die integrierte Konsole, um Ausgaben und Fehler zu sehen

```
1  System.out.println("Lecker Grillerei!")
```

```
java: ';' expected
```

Hinweise zum Programmieren

Hinweise zum Programmieren

- Achte auf korrekte Syntax (Semikolons, Klammern, etc.)

Hinweise zum Programmieren

- Achte auf korrekte Syntax (Semikolons, Klammern, etc.)
- Verwende sprechende Variablennamen (z. B. `alter` statt `x`)

Hinweise zum Programmieren

- Achte auf korrekte Syntax (Semikolons, Klammern, etc.)
- Verwende sprechende Variablennamen (z. B. `alter` statt `x`)
- Teste deinen Code regelmäßig, um Fehler frühzeitig zu finden

Autoformatter in IntelliJ IDEA

Autoformatter in IntelliJ IDEA

- Autoformatter sorgt für übersichtliche und einheitliche Code-Formatierung.

Autoformatter in IntelliJ IDEA

- Autoformatter sorgt für übersichtliche und einheitliche Code-Formatierung.
- Tastenkombination: `Strg + Alt + L` (Windows/Linux), `Cmd + Opt + L` (Mac).

Autoformatter in IntelliJ IDEA

- Autoformatter sorgt für übersichtliche und einheitliche Code-Formatierung.
- Tastenkombination: `Strg + Alt + L` (Windows/Linux), `Cmd + Opt + L` (Mac).
- Achtet auf Einrückungen, Abstände und Zeilenumbrüche.

Autoformatter in IntelliJ IDEA

- Autoformatter sorgt für übersichtliche und einheitliche Code-Formatierung.
- Tastenkombination: `Strg + Alt + L` (Windows/Linux), `Cmd + Opt + L` (Mac).
- Achtet auf Einrückungen, Abstände und Zeilenumbrüche.
- Gut formatierter Code ist leichter lesbar und verständlich.

Autoformatter in IntelliJ IDEA

- Autoformatter sorgt für übersichtliche und einheitliche Code-Formatierung.
- Tastenkombination: `Strg + Alt + L` (Windows/Linux), `Cmd + Opt + L` (Mac).
- Achtet auf Einrückungen, Abstände und Zeilenumbrüche.
- Gut formatierter Code ist leichter lesbar und verständlich.
- Tipp: Code regelmäßig vor dem Speichern oder Teilen formatieren.