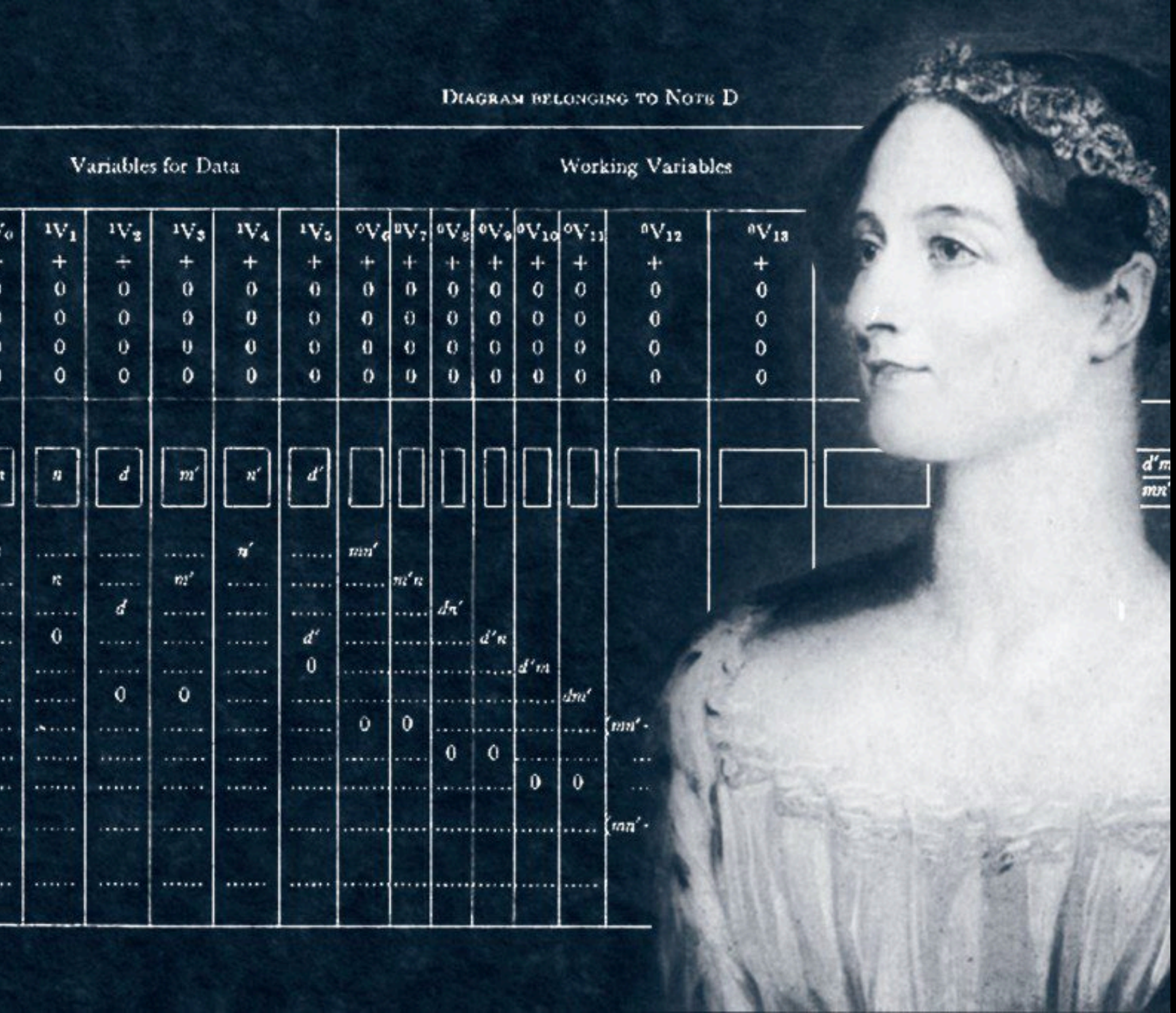


Ada Lovelace

[aus Ihren Notizen]

> „Die Grenzen der Arithmetik wurden in dem Augenblick überschritten, in dem die Idee zur Verwendung der [Programmier]Karten entstand, und die Analytical Engine hat keine Gemeinsamkeit mit schlichten Rechenmaschinen. Sie ist einmalig, und die Möglichkeiten, die sie andeutet, sind höchst interessant.“

> „[Die Analytical Engine] könnte auf andere Dinge als Zahlen angewandt werden, wenn man Objekte finden könnte, deren Wechselwirkungen durch die abstrakte Wissenschaft der Operationen dargestellt werden können und die sich für die Bearbeitung durch die Anweisungen und Mechanismen des Gerätes eignen.“

Prinzipien der Programmierung

Objektorientiere Programmierung (Java)

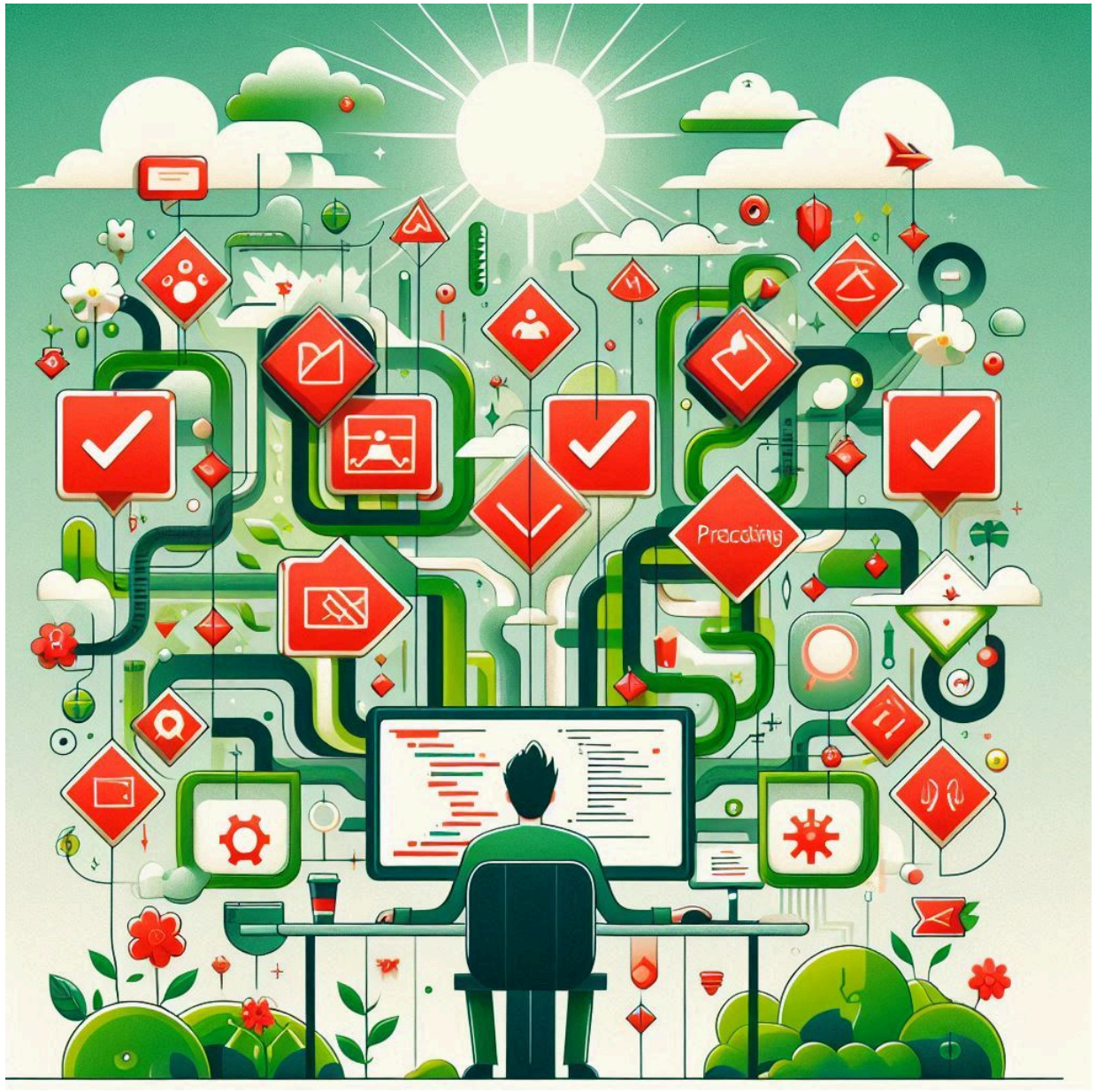
Daniel Merkle

Wintersemester 2024/2025

Prinzipien der Programmierung (PdP)

PdP - Objektorientierte Programmierung

- Zentrale Webseite (OOP): https://pdp.algochem.techfak.de/
- Folien online. Color-code:
  - weiß - Standard
  - grün - ergänzend
  - rot - look-ahead
- Artemis: https://artemis.techfak.de/



Artemis

Übung macht den Meister!

[image from Dall-E 3]

Quizzes

(nur in der Vorlesung)

Ein Beispiel: Quiz

Heterogenität der Studierenden

- Am Anfang recht wenige Prinzipien, **Fokus auf Anfänger**
- Vor allem in der Vorlesung: auch immer wieder herausfordernde Themen
- Mehr und mehr **Prinzipien der Programmierung** im Laufe der Vorlesung

immer wieder: Definition von **Lernzielen** (üblicherweise mehrfach pro 90-Minuten Einheit)

Los gehts

Lernziele

- Einige Grundlagen
- Vertrautheit mit der Entwicklungsumgebung IntelliJ IDEA
- Nutzung der Artemis-Plattform zum Testen und Einreichen von Lösungen

Einführung in Entwicklungsumgebungen

Moderne Programmierung erfolgt oft in einer IDE (Integrated Development Environment). Sie hilft Fehler zu erkennen, unterstützt bei der Strukturierung des Codes, macht Versionsverwaltung, und und und...

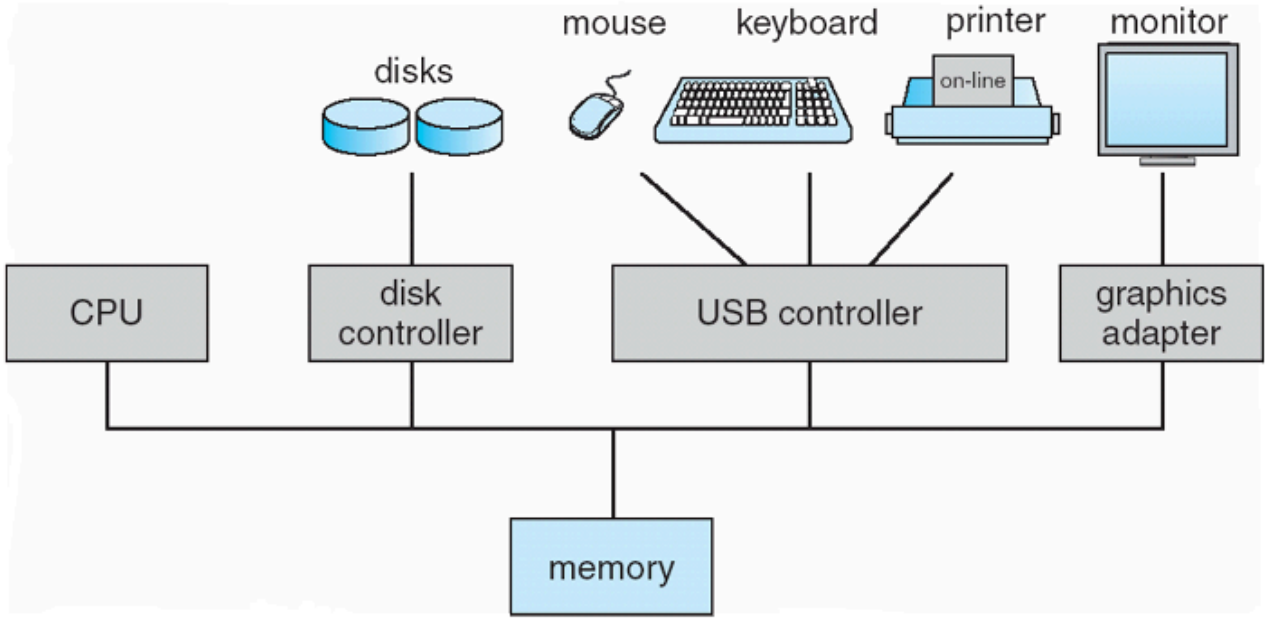
IntelliJ IDEA

Wir nutzen in diesem Kurs die IDE IntelliJ IDEA.

- Kostenlose Bildungslizenz für Studierende
- Unterstützt Java 17 JDK

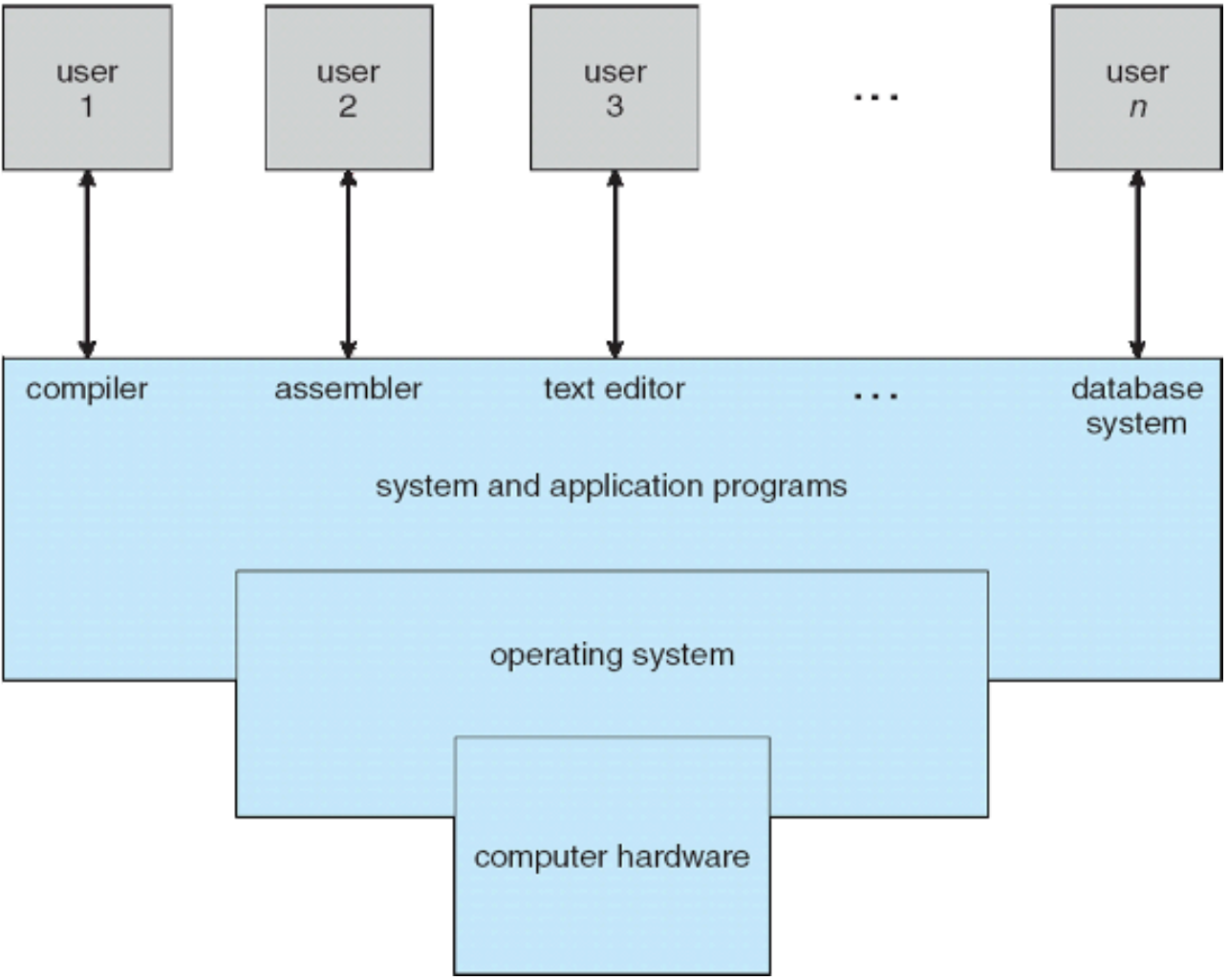
Entwicklungsumgebung ?

CLI + `vim`



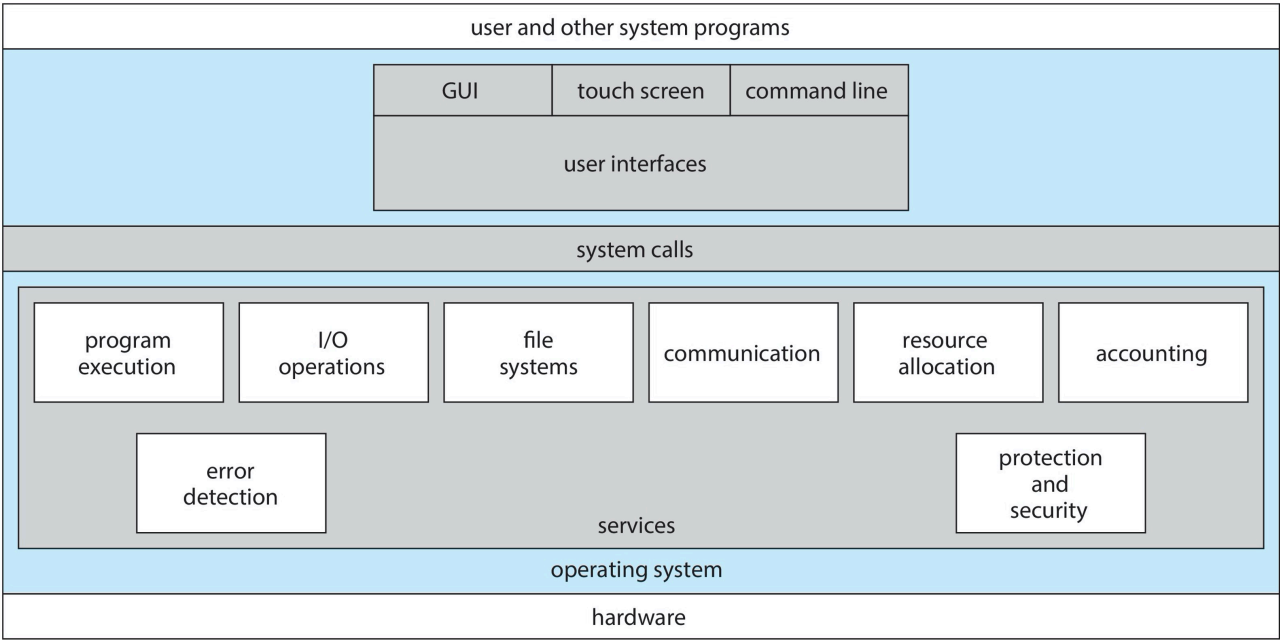
Computer System

[image from Silberschatz et al.]



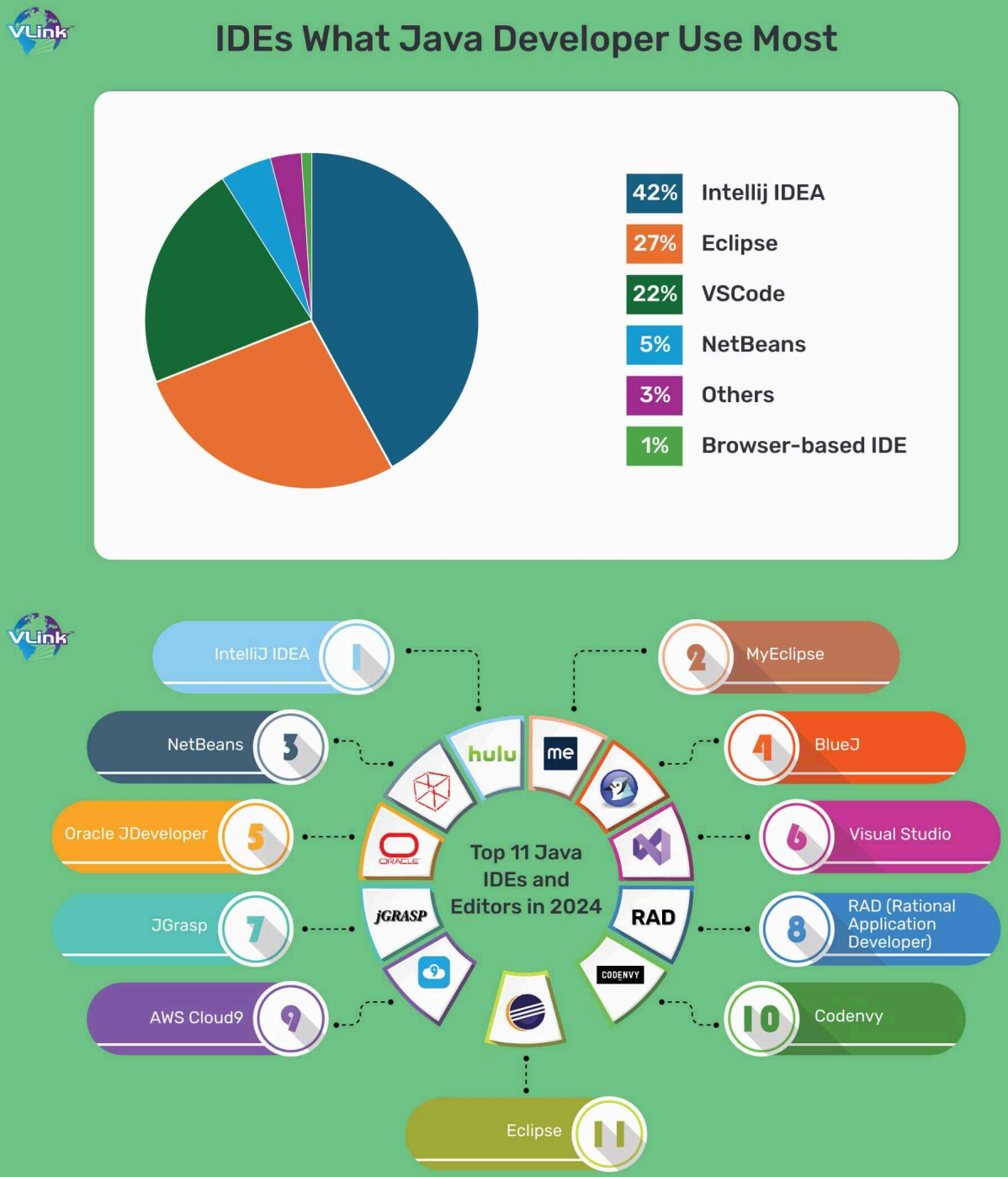
Computer System

[image from Silberschatz et al.]



Computer System

[image from Silberschatz et al.]



IDEs

[image from vlinkinfo, 2024]

Entwicklungsumgebungen

Präsentation

- CLI + `vim` / `emacs`
- code-server (VSCode)
- IntelliJ IDEA

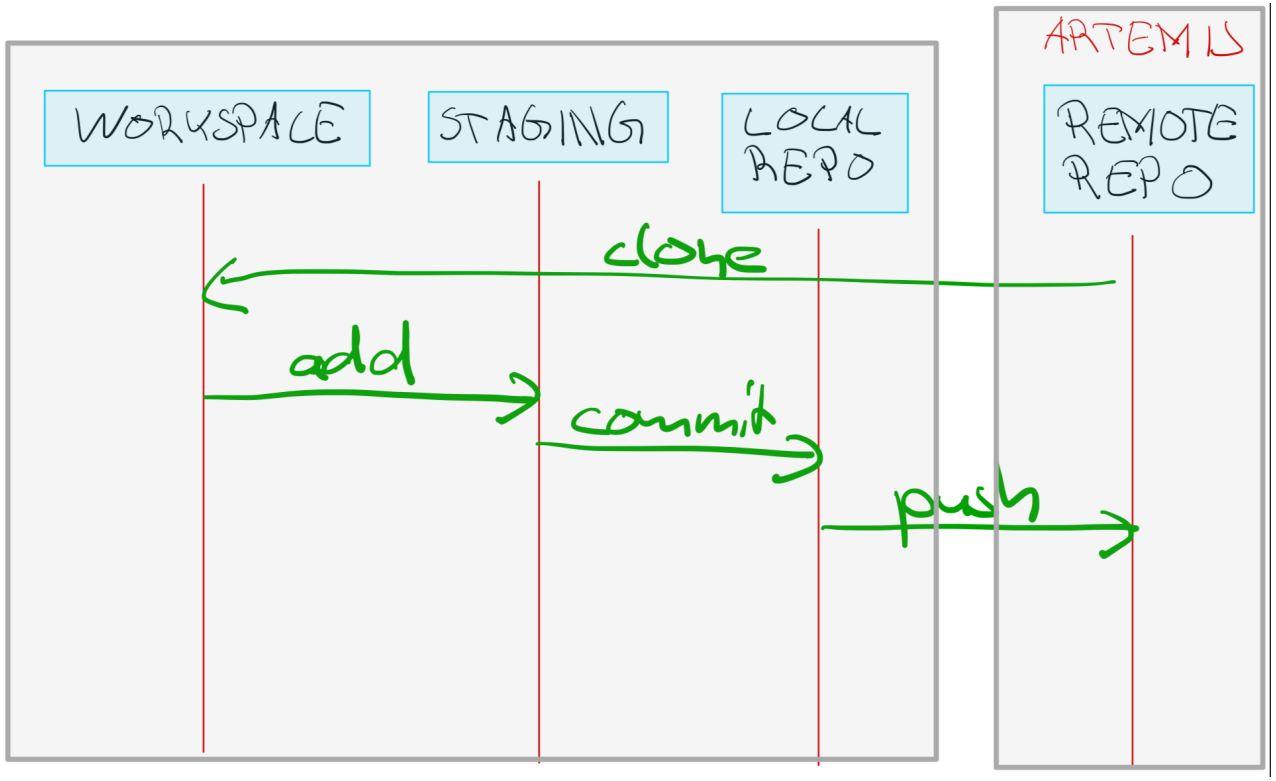
Artemis-Plattform

Die Artemis-Plattform der Technischen Fakultät wird verwendet, um Lösungen hochzuladen und zu testen.

Melden Sie sich beim *Artemis-Server* an, damit Ihr Code getestet werden kann. Nach dem erstmaligen Anmelden werden Sie innerhalb ca. eines Werktages die Übungsaufgaben des ersten Teils sehen und bearbeiten können.

Voraussetzung ist, dass Sie bei 392005 Prinzipien der Programmierung - Objektorientierte Programmierung (V) (WiSe 2024/2025) oder bei 392002 Übungen zu Prinzipien der Programmierung (Ü) (WiSe 2024/2025) angemeldet sind.

Präsentation Artemis



GIT

Git Begriffe

- **Workspace**: Ihr aktuelles Arbeitsverzeichnis mit den Dateien, an denen Sie arbeiten.
- **Staging**: Bereich, in den Sie Änderungen hinzufügen, bevor Sie `commit`en.
- **Lokales Repo**: Ihr persönliches Git-Repository auf Ihrem Rechner.
- **Remote Repo**: Ein Repository auf einem Server (z. B. Artemis).

Git Befehle:

- **clone**: Kopiert ein Remote-Repository auf deinen Rechner.
- **add**: Fügt Änderungen zum Staging-Bereich hinzu.
- **commit**: Speichert die Änderungen dauerhaft im lokalen Repository.
- **push**: Überträgt deine Commits vom lokalen ins Remote-Repository.

Artemis

`git push` triggert ein build/test
(und dadurch die Bewertung Ihrer Lösungen)

Java

Wichtige Java Versionen

Version	Jahr	Wichtige Neuerungen (Auswahl)
Java 21	2023	String Templates, Virtual Threads (finalisiert)
Java 17	2021	Pattern Matching für `switch`, neue `java.nio.file`
Java 11	2018	`var` in Lambda-Parameter, HTTP-Client API
Java 8	2014	Lambda-Ausdrücke, Stream-API
Java 7	2011	`try-with-resources`, Diamond-Operator
Java 6	2006	Scripting API, Verbesserte Diagnosemöglichkeiten

Oracle hat entschieden, dass Java 8, Java 11, Java 17, und Java 21 LTS-Versionen sind und bietet für diese längere Supportzeiträume an.

Andere Anbieter wie Adoptium, Azul, Amazon Corretto, etc., folgen oft der gleichen Praxis, indem sie die von Oracle als LTS deklarierten Versionen ebenfalls langfristig unterstützen.

Spezifikation vs. Implementierung

Java 17 Implementierungen

Spezifikation: hier

Implementierung	Firma / Konsortium	Lizenz	Wichtige Merkmale
Oracle JDK	Oracle	Proprietär, kostenlos für Tests	Offizielle kommerzielle Version, LTS-Support
OpenJDK	OpenJDK Community	GPLv2+CPE	Open-Source-Referenzimplementierung
Adoptium (Eclipse Temurin)	Eclipse Foundation	GPLv2+CPE	Vorbereitete, getestete OpenJDK-Binärdateien
Amazon Corretto	Amazon	GPLv2+CPE	Produktionsreife, plattformübergreifende JDK-Distribution
Azul Zulu	Azul Systems	GPLv2+CPE, kommerziell	Kostenlose und kommerzielle Optionen, erweiterter Support
IBM Semeru Runtime	IBM	GPLv2+CPE	Auf OpenJDK oder OpenJ9 basierende optimierte JVM
Liberica JDK	BellSoft	GPLv2+CPE	Verschiedene Pakete, einschließlich kleinerer Laufzeitversionen

Erste Schritte mit Java

Ein einfaches "Hello World" Programm in Java:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

Warum Java als Einsteigersprache durchaus kritisiert werden darf:

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

1. `public class HelloWorld`

- **public**: Dieses Schlüsselwort bedeutet, dass die Klasse von überall zugänglich ist. Andere Klassen, auch außerhalb des Pakets, können auf die `HelloWorld`-Klasse zugreifen.
- **class**: Dies ist das Schlüsselwort, das eine Klasse in Java definiert. Eine Klasse ist eine Blaupause für Objekte oder eine Sammlung von Methoden und Variablen.
- **HelloWorld**: Der Name der Klasse. In Java muss der Dateiname mit dem Klassennamen übereinstimmen. Die Datei muss also `HelloWorld.java` heißen.

2. `public static void main(String[] args)`

- **public**: Diese Methode ist öffentlich, d. h. sie kann von anderen Klassen aufgerufen werden. In diesem Fall wird die Methode vom Java-Laufzeitsystem aufgerufen, um das Programm zu starten.
- **static**: Bedeutet, dass die Methode zu der Klasse gehört und nicht zu einer Instanz der Klasse. Du brauchst also kein Objekt von `HelloWorld`, um die `main`-Methode auszuführen. Das ist wichtig, da die `main`-Methode der Einstiegspunkt für jedes Java-Programm ist und aufgerufen wird, ohne dass zuvor ein Objekt erstellt werden muss.
- **void**: Der Rückgabewert der Methode. In diesem Fall gibt die `main`-Methode nichts zurück.
- **main**: Dies ist der Name der Methode. In Java ist `main` eine spezielle Methode, die der Einstiegspunkt für alle Anwendungen ist.
- **String[] args**: Dies ist ein Parameter, der ein Array von `String`-Objekten darstellt. Wenn das Programm über die Kommandozeile aufgerufen wird, können Argumente an das Programm übergeben werden. Diese Argumente werden im `args`-Array gespeichert.

3. `System.out.println("Hello World");`

- **System**: Dies ist eine vordefinierte Klasse in Java, die den Standard-Ein- und Ausgabe-Stream (sowie andere Dinge) enthält.
- **out**: Dies ist eine statische Variable von `System`, die eine Instanz von `PrintStream` darstellt. Sie wird verwendet, um Text in die Konsole zu schreiben.
- **println()**: Dies ist eine Methode von `PrintStream`, die den übergebenen String in die Konsole ausgibt und danach einen Zeilenumbruch hinzufügt.
- **"Hello World"**: Dies ist der Text, der in die Konsole geschrieben wird. Es handelt sich um eine Zeichenkette (String).

Zusammengefasst:

- **Die Klasse** `HelloWorld` enthält eine **`main`-Methode**, die der Startpunkt des Programms ist.
- Innerhalb dieser Methode wird die Zeile `System.out.println("Hello World");` ausgeführt, die die Zeichenkette **"Hello World"** in der Konsole ausgibt.

Ablauf des Programms:

1. Die JVM (Java Virtual Machine) sucht nach der `main`-Methode in der Klasse `HelloWorld`.
2. Die `main`-Methode wird ausgeführt.
3. Der Befehl `System.out.println("Hello World");` wird aufgerufen, wodurch "Hello World" auf der Konsole ausgegeben wird.

Ausgabe des Programms:

```
Hello World
```

Aufbau eines Java-Programms

Jedes Java-Programm benötigt:

- Eine **Klasse** (z.B. `HelloWorld`)
- Eine **main**-Methode: Startpunkt des Programms
- Anweisungen in der Methode, z.B. `System.out.println()`, um etwas auszugeben

IntelliJ IDEA: Installation

1. IntelliJ IDEA von der offiziellen Website herunterladen.
2. Wählen Sie die "Ultimate" Version (mit Bildungslizenz). (Die "Community Edition" sollte auch hinreichend sein. Nutzen Sie die "Ultimate" Version für fortgeschrittenere Technologien oder Frameworks, insbesondere Webentwicklung, Datenbanken oder Unternehmensanwendungen.)
3. Java (**JDK 17**) installieren (siehe auch hier)

Abkürzungen

- IDE (Integrated Development Environment)
  Eine **IDE** ist eine Software, die Programmierern beim Schreiben, Testen und Debuggen von Code hilft. Sie bietet Werkzeuge wie einen Editor, Compiler, Debugger und oft eine grafische Benutzeroberfläche (GUI).

- SDK (Software Development Kit)
  Ein **SDK** enthält Bibliotheken, Tools und Dokumentationen, um Anwendungen für eine bestimmte Plattform oder Sprache zu entwickeln.

- JDK (Java Development Kit)
  Das **JDK** ist ein Entwickler-Toolkit für Java-Programme. Es enthält den Compiler, Debugger, und andere Tools, die zum Entwickeln und Ausführen von Java-Anwendungen notwendig sind.

- JRE (Java Runtime Environment)
  Das **JRE** (Teil der JDK) enthält alle Laufzeitkomponenten, die nötig sind, um Java-Programme auszuführen, aber keine Entwicklungswerkzeuge wie Compiler. Es wird verwendet, um bereits kompilierte Java-Anwendungen laufen zu lassen.

- IntelliJ IDEA (Integrated Development Environment for Java)
  IntelliJ IDEA ist eine beliebte IDE, die speziell für die Entwicklung von Java-Anwendungen entwickelt wurde, inzwischen aber viele andere Sprachen und Frameworks unterstützt.

- CLI (command line interface)

Erste Aufgabe: Übungsvorlage

Die Übungsvorlage enthält bereits den folgenden Code:

```
public class Sandbox {
    public static void main(String[] args) {
        System.out.println("Willkommen in der Sandbox!");
    }
}
```

Führen Sie das Programm aus und ändern Sie den Text der Ausgabe.

Quellcode und Kompilierung

- Der von Ihnen geschriebene Code wird als **Quellcode** bezeichnet.
- Java benötigt einen **Compiler**, um den Quellcode in "Maschinensprache" (für die Java Virtual Machine) zu übersetzen.
- Die IDE übernimmt dies automatisch für Sie.

Präsentation: godbolt

Code und Kommentare

In Java können Kommentare geschrieben werden, um den Code zu erklären:

```
// Dies ist ein einzeiliger Kommentar
/* Dies ist ein mehrzeiliger Kommentar */
System.out.println("Dies ist ein Beispiel für eine Ausgabe");
```

Kommentare helfen Ihnen, den Zweck bestimmter Teile des Codes zu erklären.

Java: Struktur und Syntax

Ein einfaches Java-Programm folgt immer einer klaren Struktur:

- `class` : Definiert eine Klasse
- `main` : Startpunkt des Programms
- `{}` : Markieren Blöcke von Code
- `;` : Beendet Anweisungen

Wichtige Tools und Ressourcen

- IntelliJ IDEA: Ihre IDE für diesen Kurs
- Java JDK 17: Java-Entwicklungskit
- Artemis-Plattform: Übungsplattform

Vergleich von "Hello World" in Java 17 und Java 21

Java 17:

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Java 21 mit "Unnamed Classes" und "Instance Main Methods":

```
void main() {
    System.out.println("Hello, World!");
}
```

Was ist Java-Bytecode?

- **Definition**: Java-Bytecode ist der Zwischencode, der entsteht, wenn Java-Quellcode kompiliert wird.
- **Maschinenunabhängig**: Bytecode ist nicht an eine bestimmte Hardwarearchitektur gebunden, sondern läuft auf der Java Virtual Machine (JVM).
- **Kompilierung**: Der Java-Compiler (javac) übersetzt den Java-Quellcode in Bytecode. Dieser Bytecode wird dann von der JVM ausgeführt.

Was ist eine virtuelle Maschine (VM)?

- **Java Virtual Machine (JVM)**: Eine virtuelle Maschine ist eine Softwareschicht, die auf einem Computer ausgeführt wird und den Bytecode interpretiert und ausführt.
- **Plattformunabhängigkeit**: Die JVM ermöglicht es, dass derselbe Bytecode auf verschiedenen Betriebssystemen und Hardwarearchitekturen ausgeführt werden kann.
- **Zweck**: Die JVM fungiert als Übersetzer zwischen dem plattformunabhängigen Bytecode und den spezifischen Maschinenbefehlen des jeweiligen Betriebssystems.

Was ist Assembler Code?

- **Definition:** Assembler-Code ist eine maschinennahe Programmiersprache, die direkte Befehle an die Hardware sendet.
- **Maschinensprache:** Diese wird von Prozessoren ausgeführt und ist spezifisch für eine bestimmte Hardwarearchitektur.
- **Unterschied zu Java-Bytecode:**
  - Java-Bytecode läuft auf der JVM, die plattformunabhängig ist.
  - Assembler-Code ist hardwareabhängig und für ein bestimmtes CPU-Modell optimiert.
- **Beispiel für Assembler-Code (x86):**

```
mov eax, 1
mov ebx, 0
int 0x80
```

Zusammenfassung

- Sie haben IntelliJ IDEA und Artemis kennengelernt.
- Sie haben ein einfaches "Hello World" Programm geschrieben und ausgeführt.
- Sie verstehen die grundlegende Struktur eines Java-Programms.

Ausgabe

Lernziele

- Lernen, ein Programm zu schreiben, das Text ausgibt.
- Mit der Ausführung von Programmen vertraut werden.
- Verstehen, was der Begriff "Parameter" bedeutet.

Einfacher Ausgabebefehl

Der Ausgabebefehl `System.out.println("Hello world");` gibt den Text "Hello world" aus.

```
System.out.println("Hello world!");
```

Beispielausgabe

Wenn wir den obigen Befehl ausführen, erhalten wir folgende Ausgabe:

```
Hello world!
```

Beliebiger Text

Mit dem Befehl `System.out.println("Text")` können Sie jeden gewünschten Text ausgeben. Der folgende Befehl gibt den Text "Hello there!" aus:

```
System.out.println("Hello there!");
```

Quiz

Programmrumpf

Jedes Java-Programm benötigt einen Programmrumpf, damit es funktioniert. Beispiel:

```
public class Example {
    public static void main(String[] args) {
        System.out.println("Text to be printed");
    }
}
```

Ausführungsfluss

- Das Programm startet nach der Zeile `public static void main(String[] args)`.
- Der Computer führt den Code nach der ersten Klammer `{` Zeile für Zeile aus, bis die geschweifte Klammer `}` erreicht ist.
- PythonTutor, zur Visualisierung

Was ist `System`?

`System` ist eine Klasse in Java, die Teil des Pakets `java.lang` ist. Diese Klasse stellt verschiedene nützliche Funktionen bereit, die auf den Systemzustand und die Umgebungsvariablen zugreifen.

Was ist `System.out`?

`System.out`, ist ein Objekt der Klasse `PrintStream`, das Teil der `System`-Klasse ist (als statisches Feld). Es repräsentiert den Standardausgabestrom (`stdout`), der normalerweise die Konsole oder das Terminal ist.

Was ist `System.out.println`?

`System.out.println()` ist eine Methode des `PrintStream`-Objekts, die verwendet wird, um Text an den Standardausgabestrom zu senden. Sie gibt den angegebenen Text aus und fügt automatisch einen Zeilenumbruch hinzu.

Mehrere Ausgaben

Wir können mehrere `System.out.println()` -Befehle hintereinander ausführen:

```
System.out.println("Hello");
System.out.println("World!");
```

Ausgabe:

```
Hello
World!
```

Parameter

Die Informationen, die durch den Ausgabebefehl ausgegeben werden, sind seine **Parameter**. Zum Beispiel:

```
System.out.println("Hi");
```

Hier ist `"Hi"` der Parameter.

Semikolon trennt Befehle

Befehle in Java müssen mit einem Semikolon `;` abgeschlossen werden. Mehrere Befehle können auf einer Zeile stehen:

```
System.out.println("Hello"); System.out.println("world");
```

Ausgabe:

```
Hello
world
```

Kommentare

Kommentare helfen, den Quellcode zu erklären und Notizen hinzuzufügen. Zwei Arten von Kommentaren:

- Einzeilige Kommentare: `// Kommentar`
- Mehrzeilige Kommentare: `/* Kommentar */`

Beispiel:

```
// Einzeiliger Kommentar
/*
   Mehrzeiliger Kommentar
*/
```

Beispiel: Kommentare

```
public class Comments {
    public static void main(String[] args) {
        // Ausgabe
        System.out.println("Text to print");
        /* Mehrzeiliger Kommentar
        - weitere Erklärungen
        */
        System.out.println("Anderer Text zum Ausgeben");
    }
}
```

Zusammenfassung

- Sie haben gelernt, wie man Text in Java ausgibt.
- Sie kennen die Struktur eines einfachen Java-Programms.
- Sie verstehen, wie Parameter und Kommentare funktionieren.

Eingaben lesen

Lernziele

- Ein Programm schreiben, das Texteingaben von Nutzenden liest.
- Verstehen, was ein "String" ist.
- Strings verknüpfen (konkatenieren).

Texteingabe

Eingaben beziehen sich auf Text, den Nutzende schreiben und der vom Programm gelesen wird. Eingaben werden immer als **String** gelesen. Zum Lesen verwenden wir als "Werkzeug" die `Scanner`-Klasse.

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

Was macht `import`?

Der `import`-Befehl in Java wird verwendet, um externe Klassen und Pakete in einem Programm verfügbar zu machen. Damit kann auf Funktionen und Werkzeuge zugegriffen werden, die nicht im Standardumfang des aktuellen Programms enthalten sind.

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

Hier importieren wir die Klasse `Scanner` aus dem Paket `java.util`, um Eingaben zu lesen.

Was ist ein Package?

Ein **Package** (Paket) in Java ist eine Sammlung von Klassen, Schnittstellen und Unterpaketen, die thematisch zusammengehören. Es dient dazu, Code zu organisieren und Kollisionen zwischen gleichnamigen Klassen zu vermeiden.

Beispiel:

- `java.util` : Ein Paket, das nützliche Werkzeuge wie den `Scanner` enthält.

Was ist `Scanner`?

`Scanner` ist eine Klasse aus dem Paket `java.util`, die verwendet wird, um Eingaben von Nutzenden zu lesen. Sie ermöglicht es, Texteingaben von der Konsole zu verarbeiten.

```
Scanner scanner = new Scanner(System.in);
```

Der `Scanner` erwartet als Argument eine Eingabequelle, wie `System.in` für Tastatureingaben.

Was ist eine Klasse?

Eine **Klasse** in Java ist eine Blaupause, die die Struktur und das Verhalten von Objekten definiert. Sie enthält Eigenschaften (Variablen) und Methoden (Funktionen), die von Objekten dieser Klasse verwendet werden.

Beispiel: `Scanner` ist eine Klasse, die Methoden zum Lesen von Eingaben bereitstellt.

Was ist ein Objekt?

Ein **Objekt** ist eine Instanz einer Klasse. Es wird aus der Klasse erstellt und kann deren Methoden und Eigenschaften verwenden.

```
Scanner scanner = new Scanner(System.in);
```

Hier ist `scanner` ein Objekt der Klasse `Scanner`, das die Methode `nextLine()` zum Lesen von Texteingaben verwendet.

Scanner-Werkzeug

Das Scanner-Werkzeug wird erstellt, um die Eingabe zu lesen:

```
Scanner scanner = new Scanner(System.in);
```

Das Werkzeug liest Eingaben von der Standard-Eingabequelle, d.h., der Tastatur.

`System.in` ist ein Eingabestream in Java, der für das Einlesen von Daten von der Standard-Eingabequelle verwendet wird, normalerweise die Tastatur (Konsole).

Beispiel: Einfache Eingabe

Ein einfaches Beispiel für ein Programm, das nach einer Eingabe fragt und diese ausgibt:

```
import java.util.Scanner;

public class Program {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Write a message: ");
        String message = scanner.nextLine();

        System.out.println(message);
    }
}
```

Beispielausgabe

Ausgabe:

Write a message:

```
Hello!
```

Hello!

Was ist ein Scanner?

Ein `Scanner` ist ein Werkzeug, das Daten von der Konsole liest. Es ist Teil des Pakets `java.util` und ermöglicht es dem Programm, Eingaben von Nutzenden zu verarbeiten.

Was ist ein String?

Ein **String** ist eine Zeichenkette, die aus mehreren Zeichen besteht. In Java wird ein String durch doppelte Anführungszeichen `" "` gekennzeichnet.

```
String message = "Hello!";
```

Strings verknüpfen (Konkatenation)

Strings können mit dem `+`-Operator miteinander verknüpft werden.

```
String firstName = "Ada";
String lastName = "Lovelace";

String fullName = firstName + " " + lastName;
```

Das Ergebnis ist die Verkettung der beiden Strings.

Mehrfache Eingaben

Ein Programm kann mehrfach Eingaben abfragen und die Strings kombinieren:

```
System.out.println("What is your first name?");
String firstName = scanner.nextLine();

System.out.println("What is your last name?");
String lastName = scanner.nextLine();

String fullName = firstName + " " + lastName;
System.out.println("Your full name is: " + fullName);
```

Beispielausgabe

Ausgabe:

```
What is your first name?
```

Ada

```
What is your last name?
```

Lovelace

```
Your full name is: Ada Lovelace
```

Übung: Geschichte

Schreiben Sie ein Programm, das nach dem Namen und dem Beruf einer Person fragt und eine Geschichte ausgibt.

Beispielausgabe: Geschichte

```
I will tell you a story, but I need some information first.
What is the main character called?
```

Bob

```
What is their job?
```

a builder

```
Here is the story:
Once upon a time there was Bob, who was a builder.
On the way to work, Bob reflected on life.
Perhaps Bob will not be a builder forever.
```

Zusammenfassung

- Sie haben gelernt, wie Sie Eingaben von Nutzenden lesen.
- Sie verstehen, was ein String ist und wie Sie Strings verknüpfen.
- Sie haben ein Beispiel für die Verwendung von Strings in einer Geschichte gesehen.

Variablen in Java

Lernziele

- Verstehen, was Variablen sind und wie sie verwendet werden.
- Verschiedene Variablentypen kennenlernen (z.B. `int`, `double`, `boolean`).
- Variablen deklarieren, initialisieren und verwenden.

Was ist eine Variable?

Eine **Variable** ist ein benannter Speicherbereich, der einen bestimmten Datentyp speichern kann. Jede Variable hat:

- Einen **Typ** (z.B. `int`, `double`, `boolean`)
- Einen **Namen**
- Einen **Wert**

```
int number = 123;
double floatingPoint = 3.1415;
boolean isTrue = true;
```

Beispiel: Variablen

Ein Beispielprogramm, das Variablen verschiedener Typen deklariert und ausgibt:

```
String text = "contains text";
int wholeNumber = 123;
double floatingPoint = 3.141592653;
boolean trueOrFalse = true;

System.out.println("Text variable: " + text);
System.out.println("Integer variable: " + wholeNumber);
System.out.println("Floating-point variable: " + floatingPoint);
System.out.println("Boolean: " + trueOrFalse);
```

Beispielausgabe

Ausgabe:

```
Text variable: contains text
Integer variable: 123
Floating-point variable: 3.141592653
Boolean: true
```

Variablen neu zuweisen

Eine Variable kann neu zugewiesen werden, nachdem sie deklariert wurde. Beispiel:

```
int value = 10;
System.out.println(value);
value = 4;
System.out.println(value);
```

Beispielausgabe

Ausgabe:

```
10
4
```

Variablen und Typen

Der Variablentyp muss beim Erstellen der Variable deklariert werden. Wenn ein neuer Wert zugewiesen wird, darf der Typ nicht mehr angegeben werden.

```
int value = 10;   // Deklaration und Initialisierung
value = 4;        // Zuweisung eines neuen Wertes
```

Was ist eine Variable in Java?

In Java ist eine **Variable** ein benannter Bereich im Speicher, der einen Wert speichert. Jede Variable hat einen **Datentyp**, der den Typ der Informationen bestimmt, die gespeichert werden können (z.B. `int` für Ganzzahlen).

Verschiedene Variablentypen in Java

- **int**: Ganzzahlen (z.B. `123`)
- **double**: Gleitkommazahlen (z.B. `3.14`)
- **boolean**: Wahrheitswerte (`true` oder `false`)
- **String**: Zeichenketten (z.B. `"Hallo"`)

Jeder Typ speichert einen anderen Typ von Daten.

Variablen sind eindeutig

Jede Variable muss einen eindeutigen Namen haben. Es ist nicht erlaubt, zwei Variablen mit demselben Namen zu deklarieren.

```
double pi = 3.14;
double pi = 3.141592653;  // Fehler: doppelte Deklaration
```

Quiz

Quiz Kopieren

Achtung!

```java
public class FloatingPointLimitations {
    public static void main(String[] args) {
        // Beispiel Präzisionsverlust
        double a = 0.1;
        double b = 0.2;
        double c = 0.3;

        if (a + b == c) {
            System.out.println("a + b ist genau gleich c");
        } else {
            System.out.println("a + b ist nicht genau gleich c");
        }
    }
}
```

Gleitkommazahlen auf Gleichheit zu prüfen ist eine schlechte Idee!

Zusammenfassung

- Sie haben gelernt, wie Variablen deklariert und verwendet werden.
- Sie kennen die grundlegenden Variablentypen in Java.
- Sie haben gesehen, wie Variablen neu zugewiesen werden können.

Quiz 127

Quiz 128

Berechnungen in Java

Lernziele

- Verstehen, wie Berechnungen in Java durchgeführt werden.
- Grundlegende Operatoren kennenlernen (Addition, Subtraktion, Multiplikation, Division).
- Kombination von Variablen und Ausdrücken.

Mathematische Operationen

In Java können einfache Berechnungen mit Operatoren durchgeführt werden:

- Addition: `+`
- Subtraktion: `-`
- Multiplikation: `*`
- Division: `/`

Beispiel:

```
int sum = 5 + 3;
int difference = 10 - 2;
int product = 4 * 3;
double quotient = 8.0 / 2.0;
```

Beispiel: Berechnungen

Ein Beispielprogramm, das Berechnungen mit Variablen durchführt:

```
int first = 5;
int second = 23;

int sum = first + second;
int difference = second - first;

System.out.println("Sum: " + sum);
System.out.println("Difference: " + difference);
```

Beispielausgabe

Ausgabe:

```
Sum: 28
Difference: 18
```

Kombination von Variablen und Text

In Java können Sie Berechnungen und Texte kombinieren:

```
System.out.println("Four: " + (2 + 2));
System.out.println("But! Twenty-two: " + 2 + 2);
```

Beispielausgabe

Ausgabe:

```
Four: 4
But! Twenty-two: 22
```

Was sind Operatoren in Java?

Operatoren sind Symbole, die Operationen auf Variablen und Werten durchführen. In Java gibt es viele Arten von Operatoren:

- Arithmetische Operatoren: `+`, `-`, `*`, `/`
- Vergleichsoperatoren: `==`, `!=`, `>`, `<`

Was ist der Unterschied zwischen `2 + 2` und `"2" + "2"`?

- `2 + 2` ist eine **Addition**, die den Wert `4` ergibt.
- `"2" + "2"` ist eine **Konkatenation** von Strings, die den Text `"22"` ergibt.

Java interpretiert den `+`-Operator anders, je nachdem, ob die Operanden Zahlen oder Zeichenketten sind.

Modulo-Operator

Der Modulo-Operator `%` gibt den Rest einer Division zurück:

```
int remainder = 10 % 3;
System.out.println("Remainder: " + remainder);
```

Beispielausgabe

Ausgabe:

```
Remainder: 1
```

Code Visualization 2+2, 5-calculating

Code Visualization Schrittweise, 5-calculating

Quizzes

Quiz Variablen

Quiz nochmal Variablen

Quiz Division

Zusammenfassung

- Sie haben gelernt, wie Sie Berechnungen in Java durchführen.
- Sie kennen die wichtigsten mathematischen Operatoren.
- Sie haben den Unterschied zwischen Zahlen und Strings beim Verwenden des `+` -Operators verstanden.

Bedingte Anweisungen in Java

Lernziele

- Verstehen, was eine bedingte Anweisung ist.
- Vergleichsoperatoren kennenlernen (`<`, `>`, `==`, etc.).
- Bedingungen mit `if` und `else` verwenden.

Bedingte Anweisungen (If-Statements)

Mit einer **bedingten Anweisung** können Programme Entscheidungen basierend auf Bedingungen treffen. Syntax:

```
if (Bedingung) {
    // Code wird ausgeführt, wenn die Bedingung wahr ist
}
```

Beispiel: Einfache Bedingung

Ein einfaches Beispiel für eine bedingte Anweisung:

```
int age = 18;

if (age >= 18) {
    System.out.println("You are an adult.");
}
```

Beispielausgabe

Ausgabe:

```
You are an adult.
```

Vergleichsoperatoren

Java unterstützt verschiedene **Vergleichsoperatoren** zur Prüfung von Bedingungen:

- `==` : gleich
- `!=` : ungleich
- `<` : kleiner
- `<=` : kleiner oder gleich
- `>` : größer
- `>=` : größer oder gleich

Beispiel:

```
if (a > b) {
    System.out.println("a is greater than b");
}
```

Beispiel: If-Else

Zusätzlich zu `if` kann man eine **else-Anweisung** verwenden, um alternative Aktionen durchzuführen:

```
int number = 10;

if (number > 0) {
    System.out.println("Positive number");
} else {
    System.out.println("Non-positive number");
}
```

Beispielausgabe

Ausgabe:

```
Positive number
```

Verschachtelte Bedingungen

Bedingte Anweisungen können auch verschachtelt werden:

```
int number = 0;

if (number > 0) {
    System.out.println("Positive number");
} else if (number == 0) {
    System.out.println("Zero");
} else {
    System.out.println("Negative number");
}
```

Beispielausgabe

Ausgabe:

```
Zero
```

Was ist eine bedingte Anweisung?

Eine **bedingte Anweisung** ist ein Codeblock, der nur ausgeführt wird, wenn eine bestimmte Bedingung wahr ist. Diese Bedingungen werden mit Vergleichsoperatoren geprüft.

Was ist ein Vergleichsoperator?

Ein **Vergleichsoperator** ist ein Symbol, das zwei Werte vergleicht, um eine Bedingung zu prüfen. Beispiele:

- `==` vergleicht, ob zwei Werte gleich sind.
- `>` prüft, ob ein Wert größer ist als ein anderer.

Boolesche Variablen

Eine **boolesche Variable** speichert entweder `true` oder `false`. Diese Werte werden oft in bedingten Anweisungen verwendet.

```
boolean isTrue = true;

if (isTrue) {
    System.out.println("This is true!");
}
```

Beispielausgabe

Ausgabe:

```
This is true!
```

Code Visualization Bedingte Anweisung, 6-conditional-statements

Ohne Worte

```java
public class UnleserlichesBeispiel {
    public static void main(String[] args) {
        int a = 42, b = 7, c = -3, d = 0;

        if ((((a++ / --b * (c - a) % (d + c)) == (a & b | c ^ d)) &&
            (!((a == b) || (c != d))) && ((a << 2) >= (b >> 1))) ||
            (((a + b - c * d / a) != (b % c + d)) &&
            ((a & ~b | c ^ d) == (a++ - --b)))) {

            if (!(a > b)) {
                if ((d + c - a * b) != (a / b % c)) {
                    System.out.println("Ausdruck ist wahr");
                } else if ((a == c) && ((b != d) || (c >= a))) {
                    System.out.println("Alternativer Ausdruck ist wahr");
                } else {
                    System.out.println("Keine Bedingung erfüllt");
                }
            } else {
                System.out.println("Äußere Bedingung nicht erfüllt");
            }
        } else {
            System.out.println("Ausdruck ist falsch");
        }
    }
}
```

Java Operatoren: AND, OR, XOR, NOT

Operator / Bedeutung	Syntax	Beispiel
Logisches UND	`&&`	`if (a && b) { /* Code */ }`
Logisches ODER	`||`	`if (a || b) { /* Code */ }`
Logisches NICHT	`!`	`if (!a) { /* Code */ }`
Bitweises UND	`&`	`int c = a & b;`
Bitweises ODER	`|`	`int c = a | b;`
Bitweises XOR	`^`	`int c = a ^ b;`
Bitweises NICHT	`~`	`int c = ~a;`

Java Vergleichsoperatoren: AND, OR, NOT

Operator / Bedeutung	Syntax	Beispiel
Gleich	`==`	`if (a == b) { /* Code */ }`
Ungleich	`!=`	`if (a != b) { /* Code */ }`
Größer als	`>`	`if (a > b) { /* Code */ }`
Kleiner als	`<`	`if (a < b) { /* Code */ }`
Größer oder gleich	`>=`	`if (a >= b) { /* Code */ }`
Kleiner oder gleich	`<=`	`if (a <= b) { /* Code */ }`

Hinweise:

- **Logische Operatoren** (`&&` , `||` , `!`) werden mit `boolean` -Werten verwendet und ergeben einen `boolean` -Wert.
- **Bitweise Operatoren** (`&` , `|` , `^` , `~`) werden mit ganzzahligen Datentypen verwendet und führen bitweise Operationen durch.
- **Vergleichsoperatoren** (`==` , `!=` , `<` , `>` , `<=` , `>=`) vergleichen zwei Operanden und ergeben einen `boolean` -Wert.

Zusammenfassung

- Sie haben gelernt, wie Bedingungen in Java mit `if` und `else` überprüft werden.
- Sie kennen die wichtigsten Vergleichsoperatoren.
- Sie wissen, wie man boolesche Variablen in Bedingungen verwendet.

Programmierung in unserer Gesellschaft

Abhängigkeit von Software

Unsere Gesellschaft ist stark von der von Entwicklern produzierten Software abhängig. Ohne Software wären viele Dinge wie Kommunikation, Einkaufen oder Reisen viel komplizierter.

Beispiele:

- Mobiltelefone
- Kreditkarten und Online-Banking
- Online-Buchungen
- Gesundheitsdienste

Komplexität von Software

Viele der Dienstleistungen, die wir nutzen, basieren auf komplexen Software-Systemen, die hinter den Kulissen arbeiten. Ein einfaches Beispiel ist das Einchecken für einen Flug, bei dem viele Systeme miteinander kommunizieren, um den Vorgang abzuschließen.

Beispiel: Flugreise

Wenn Sie sich online für einen Flug einchecken:

1. Ihre Daten werden überprüft (Pass, Visum, Flugstatus).
2. Ihre Sitzplatzreservierungen und Kundenstatus werden verarbeitet.
3. Informationen wie der Treibstoffbedarf des Flugzeugs werden aktualisiert.

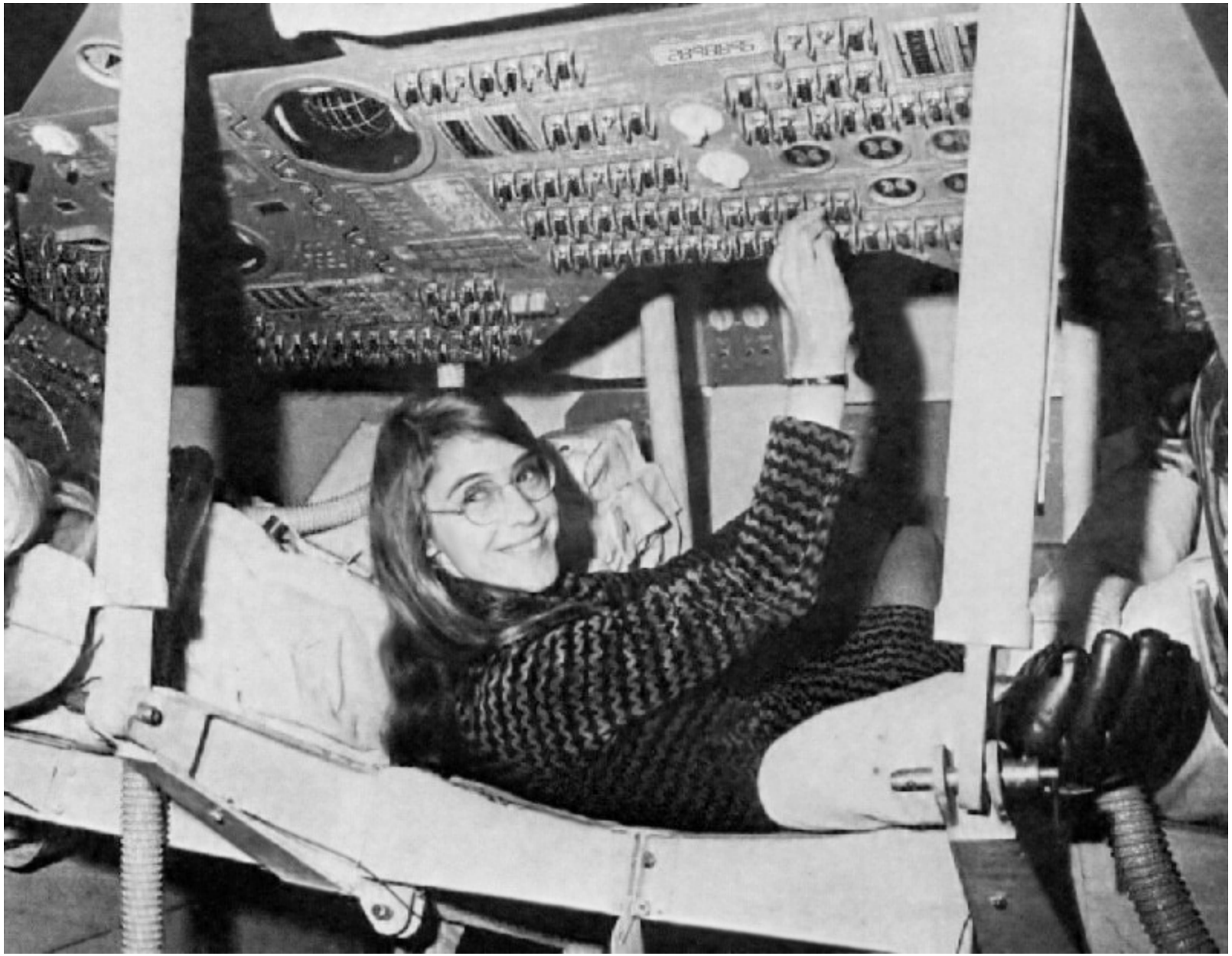
All dies geschieht durch das Klicken auf „Senden“.

Die Rolle der Softwareentwickler

Softwareprofis sind die Architekten dieser digitalen Dienste. Es ist ihre Aufgabe, diese Systeme so zu gestalten, dass sie sicher und benutzerfreundlich sind.

Unsichtbare Architekten

Die Endbenutzer wissen selten, wer hinter den Systemen steht. Ein Beispiel ist **Margaret Hamilton**, die das Programm schrieb, das die Mondlandung ermöglichte. (Die Entwicklungen Hamiltons und ihres Teams am MIT verhinderten 1969 den Abbruch der Apollo-11-Mondlandung).



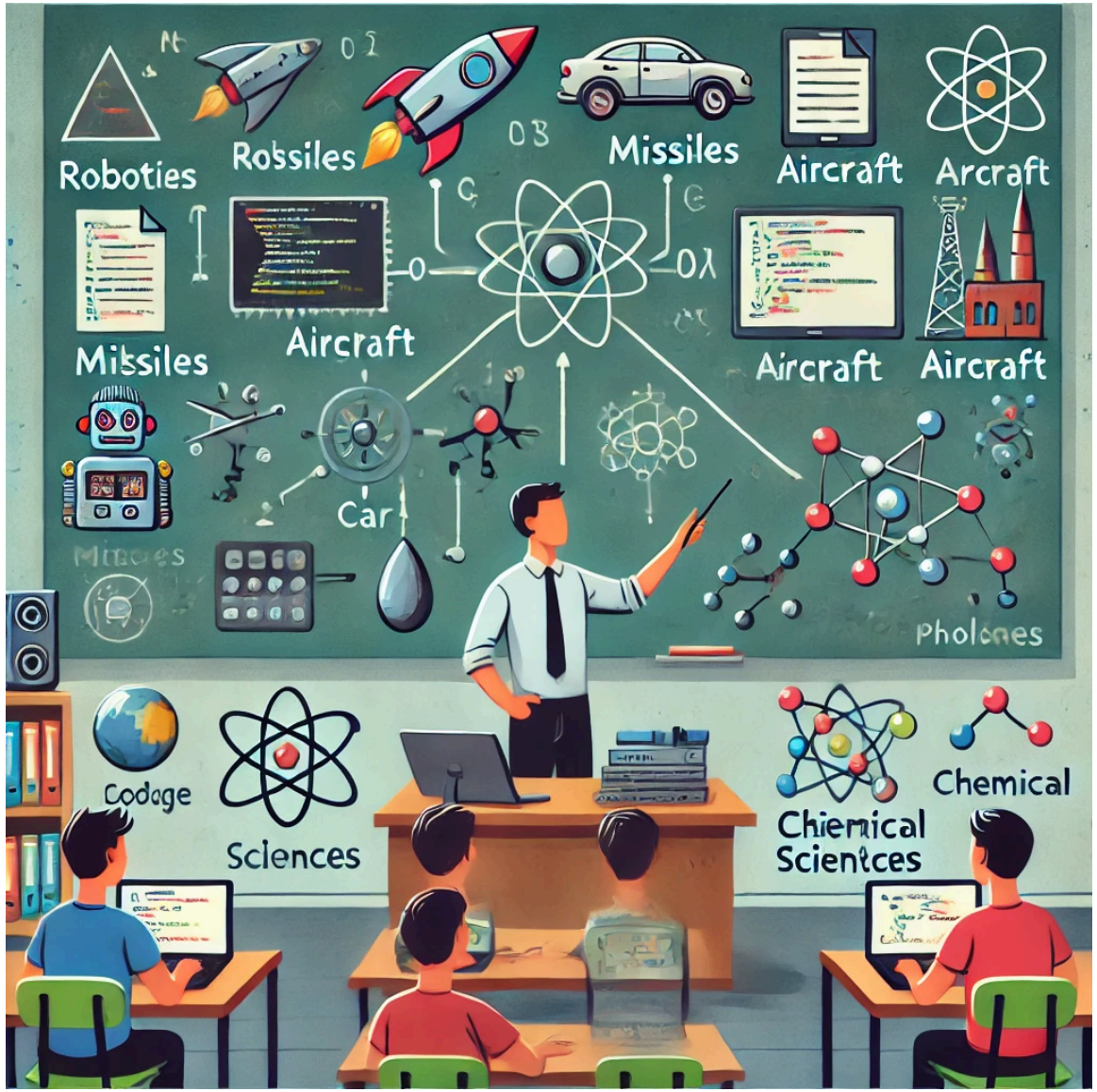
Programmieren: Die Handwerkskunst unserer Zeit

Programmieren kann als Handwerkskunst der modernen Zeit angesehen werden. Der Bedarf an Verständnis für Software und digitale Systeme wächst ständig.

Programmieren in verschiedenen Bereichen

Programmieren ist nicht nur auf Informatik beschränkt. Viele Berufe nutzen Software:

- Meteorologen, Physiker, Chemiker schreiben oft Code.
- Pädagogen nutzen digitale Systeme zur Unterstützung des Lehrens und Lernens.



ChatGPT

`Draw an image that should teach students that learn to program the importance of programming. Make illustrative examples, e.g. landing on the moon, AI, any mobile phone, chemistry ... unfortunately also in missiles. Software is everywhere.`

Beeindruckend, aber warum sind da Rechtschreibfehler?

Programmieren lernen

Grundkenntnisse des Programmierens werden bereits in Grundschulen gelehrt. Sie haben nun Ihre ersten Schritte gemacht und lernen, wie Programme funktionieren und wie man sie schreibt.

Programmieren als neue Perspektive

Nach diesem Kurs werden Sie die Welt aus der Sicht eines Programmierers betrachten. Sie werden verstehen, wie einfache Anweisungen, wie „Kaufen Sie zwei Packungen Milch“, von einem Computer interpretiert werden könnten.

Willkommen auf Ihrer Reise!

Sie sind jetzt Teil dieser Reise und es ist spannend, dass Sie dabei sind!

Zusammenfassung

- Programmierung spielt eine zentrale Rolle in unserer Gesellschaft.
- Softwareentwickler sind die Architekten der digitalen Welt.
- Die Nachfrage nach Softwarekenntnissen wächst in vielen Bereichen.
- Sie haben den ersten Schritt in die Welt des Programmierens gemacht.