**Java Basics**

**Aufgabe 1 – Java-Programmaufbau**

Skizziere die grundlegende Struktur eines Java-Programms. Erkläre jede Komponente jeweils mit einem Satz.

**package**: Wird verwendet, um Namenskonflikte zu vermeiden und Struktur zu schaffen.

**Klasse**: Als grundlegender Bestandteil von OOP muss mindestens eine (mitsamt main-Methode) existieren und sofern nicht nur statisch, auch Attribute und weitere Methoden.

(Packages und Klassen können oder müssen teils importiert werden.)

**main**-Methode: Einstiegspunkt ins Programm, welche von der JVM zuerst aufgerufen wird, dabei ist die Signatur *public static void main(String[] args)* verpflichtend.

**Methoden** (allgemein): Objekte können miteinander interagieren.

**Variablen**: Speichert Werte und stellt sie zur Verfügung.

**Datentypen**: Geben vor was Variablen beinhalten oder Methoden zurückgeben.

**Dokumentation** (Kommentare): Macht den Code nachvollziehbarer.

**Aufgabe 2 – Einfaches Kompilieren von Java-Programmen**

Schreibe, kompiliere und führe ein einfaches Java-Programm aus, das "Hallo Welt!" ausgibt. Befolge die untenstehenden Schritte und beantworte die zugehörigen Fragen.

1. **Schreibe das Programm:**

* Erstelle eine Java-Klasse namens *HelloWorld*.
* Füge eine main-Methode hinzu, die den Text "Hallo Welt!" auf der Konsole ausgibt.

1. **Kompiliere das Programm:**

* Verwende den javac-Befehl, um das Programm zu kompilieren.
* Beschreibe den Prozess und notiere die Kompilierungsbefehle, die du verwendet hast.  
  Öffne CMD oder Terminal (in IntelliJ)  
  PS C:\Users\DLabo\IdeaProjects\sic-juc-3a> cd Unterricht\src\\_20250107Wiederholung811

PS C:\Users\DLabo\IdeaProjects\sic-juc-3a\Unterricht\src\\_20250107Wiederholung811> javac HelloWorld.java

1. **Führe das Programm aus:**

* Verwende den java-Befehl, um die kompilierte .class-Datei auszuführen.
* Beschreibe den Prozess und notiere die Ausführungsbefehle, die du verwendet hast.  
  PS C:\Users\DLabo\IdeaProjects\sic-juc-3a\Unterricht\src\\_20250107Wiederholung811> java HelloWorld

Fehler: Hauptklasse HelloWorld konnte nicht gefunden oder geladen werden

Ursache: java.lang.NoClassDefFoundError: HelloWorld (wrong name: \_20250107Wiederholung811/HelloWorld)

**Man muss nach dem Kompilieren in den src zurück mir cd**

**Dann nur das java packagefolder.Classfile (Achte auf eine aktuelle JDK-Version)**

**PS C:\Users\DLabo\IdeaProjects\sic-juc-3a> cd C:\Users\DLabo\IdeaProjects\sic-juc-3a\Unterricht\src\\_20250107Wiederholung811**

**PS C:\Users\DLabo\IdeaProjects\sic-juc-3a\Unterricht\src\\_20250107Wiederholung811> cd C:\Users\DLabo\IdeaProjects\sic-juc-3a\Unterricht\src**

**PS C:\Users\DLabo\IdeaProjects\sic-juc-3a\Unterricht\src> java \_20250107Wiederholung811.HelloWorld**

**Hello World**

**PS C:\Users\DLabo\IdeaProjects\sic-juc-3a\Unterricht\src>**

**Aufgabe 3 – Klassen und Objekte**

Fülle die Lücken passend:

Objekt

Ein Objekt ist ein spezielles, individuelles Exemplar einer Klasse, das durch spezifische Werte seiner Instanz (der Klasse) gekennzeichnet ist.

Klasse

Eine Klasse beschreibt die Struktur und das Verhalten einer Menge gleichartiger Objekte.

Lö

Lösung unter: <https://www.mosbach.dhbw.de/fileadmin/user_upload/dhbw/redaktion/mitarbeiter/neuendorf/Java1_OO_Grundlagen.pdf> Folie 8

**Aufgabe 4 - OOP**

Erkläre das Konzept der **Objektorientierten Programmierung** (OOP):

**Aufgabe 5 – Prinzipien der OOP**

Erkläre in ein bis zwei Sätzen, was die nachfolgenden Prinzipien der OOP bedeuten. Gib jeweils ein Beispiel an.

|  |  |  |
| --- | --- | --- |
|  | Erklärung | Beispiel |
| Abstraktion (Abstraction) |  |  |
| Kapselung (Encapsulation) |  |  |
| Vererbung (Inheritance) |  |  |
| Polymorphismus (Polymorphism) |  |  |

Lösung unter: [https://www.mi.fu-berlin.de/wiki/pub/ABI/AlDaBiWS13Praktikum/02\_WS13\_OOP\_Templates.pdf](https://www.mi.fu-berlin.de/wiki/pub/ABI/AlDaBiWS13Praktikum/02_WS13_OOP_Templates.pdf%20%20)  Folie 6

**Aufgabe 6 – Anwendung der Prinzipien der OOP**

Erstelle ein Java-Programm, das die vier grundlegenden Prinzipien der Objektorientierten Programmierung (OOP) demonstriert: Kapselung, Vererbung, Polymorphismus und Abstraktion. Verwende dafür das Beispiel eines Fahrzeug-Systems.

**Anforderungen:**

1. **Kapselung (Encapsulation)**:

Implementiere eine Klasse *Fahrzeug* mit privaten Attributen marke und geschwindigkeit. Stelle öffentliche Methoden (getter und setter) bereit, um auf diese Attribute zuzugreifen und sie zu ändern.

1. **Vererbung (Inheritance)**:

Erstelle eine Klasse *Auto*, die von *Fahrzeug* erbt und ein zusätzliches Attribut anzahlTueren hat.

1. **Polymorphismus (Polymorphism)**:

Erstelle eine Methode start() in der Klasse *Fahrzeug* und überschreibe diese Methode in der Klasse *Auto*, um spezifische Informationen anzuzeigen.

1. **Abstraktion (Abstraction)**:

Erstelle eine abstrakte Klasse *Mitarbeiter* mit einer abstrakten Methode berechneGehalt(). Erstelle zwei Unterklassen *VollzeitMitarbeiter* und *TeilzeitMitarbeiter*, die die Methode berechneGehalt() implementieren.

**Aufgabe 7 – Unterscheidung zwischen static und instance**

1. Worin besteht der Unterschied zwischen static und instance?

Alles, was static ist, gehört zur Klasse (Variablen und Methoden) und kann ohne Objekt, sondern stattdessen über den Klassennamen aufgerufen werden (Klasse.methode()).

Instanzen sind Objekte der Klasse, diese können eigene Attribute und Methoden haben. Objekte können auch auf statische Elemente zugreifen. Die Syntax Objekt.methodeDerKlasseStatic() wird aber so nicht empfohlen. Besser Klasse.methode(Instanzvariable).

Analogie (Beispiel Hakan KI) Mehrparteienhaus mit mehreren Wohnungen und Gemeinschafträumen (Schuppen, Keller, etc), dabei sind Bauplan = Klassen, Wohnung = Objekt.

1. Implementiere folgendes Java-Programm:
2. Implementiere eine Klasse *Rechner*, die eine statische Variable pi und eine statische Methode berechneKreisumfang(double radius) enthält.
3. Implementiere in der Klasse *Rechner* eine Instanz-Variable ergebnis und eine Instanz-Methode addiere(double a, double b).
4. Implementiere eine statische innere Klasse *MathUtils* in der Klasse Rechner, die eine statische Methode multiplikation(double a, double b) enthält.
5. Erstelle in der main-Methode ein Objekt der Klasse Rechner und demonstriere den Zugriff auf die statischen und Instanz-Mitglieder sowie die statische Klasse.