Hochschule Karlsruhe
University of Applied Sciences

Fakultät für **Informatik**
und Wirtschaftsinformatik

Prof. Dr. Carsten Sinz
Lehrgebiet Software-Entwicklung

Übungsblatt 1 zur Vorlesung Programmieren (fällig bis 19.10.25)

Hinweise (diese gelten auch für zukünftige Übungen):

- Die Abgabe der Aufgabenblätter findet (als Team) in den Übungsgruppen statt (beim Tutor vorzeigen und erklären). Nach der Abnahme sind die Lösungen noch im ILIAS-System hochzuladen. All dies spätestens in der Woche, an deren Ende das Blatt fällig ist.
- Nicht abgenommene Aufgaben können überarbeitet und eine Woche später nochmals abgegeben (und danach hochgeladen) werden.
- Verwenden Sie für Ihre Abgabe in ILIAS eine ZIP-Datei, die alle Dateien Ihrer Lösung enthält. Verwenden Sie Unterverzeichnisse für jede Aufgabe. Für handschriftliche Abgaben können Sie Ihre Lösung einscannen oder abfotografieren.
- Aufgaben sind, sofern nicht explizit bei der Aufgabe anders vermerkt, in Dreiergruppen zu bearbeiten und abzugeben.
- Bei Fragen zu den Aufgaben stehen Ihnen die Tutoren gerne zur Verfügung. Auch ein Blick in die Diskussionsgruppen bei MatterMost ist sehr zu empfehlen.
- Beginnen Sie rechtzeitig, d.h. schon vor dem Termin Ihrer Übungsgruppe, damit, sich mit den Aufgaben auseinanderzusetzen und mit der Bearbeitung zu beginnen.
- Viel Spaß und Erfolg!

Aufgabe 1 (Installation JDK): Installieren Sie das Java Development Kit (JDK) für Java Version 21 sowie einen Texteditor auf Ihrem Rechner.

Je nach Betriebssystem unterscheidet sich die Installation geringfügig.

Wenn Sie auf einem Poolrechner arbeiten wollen, erübrigt sich die Installation, JDK 21 und der Texteditor `Notepad++` sind bereits vorhanden. Sie können dann den nächsten Schritt überspringen und unten bei Teilaufgabe a) fortsetzen.

Windows und MacOS: Besuchen Sie die Seite `https://adoptium.net` und laden von dort das JDK, Version 21 "Temurin" herunter. Für Windows empfiehlt sich die Verwendung der `.msi`-Datei, für MacOS die der `.pkg`-Datei. Installieren Sie dann das JDK wie unter Ihrem Betriebssystem üblich.

Linux: Unter Linux können Sie das JDK ebenfalls von der Adoptium-Seite herunterladen. Einfacher ist allerdings, den Paketmanager Ihres Linux-Systems zur Installation des JDK 21 (hier in der OpenJDK-Variante) zu verwenden. Unter Ubuntu lässt sich dies z.B. mit

```
sudo apt install openjdk-21-jdk
```

bewerkstelligen.

Testen Sie Ihre Installation durch Eingabe von `java -version`. Die Ausgabe sollte ähnlich aussehen wie:

```
openjdk version "21.0.2"2024-01-16 LTS
OpenJDK Runtime Environment Temurin-21.0.2+13 (build 21.0.2+13-LTS)
OpenJDK 64-Bit Server VM Temurin-21.0.2+13 (build 21.0.2+13-LTS, mixed mode)
```

Installieren Sie auf Ihrem System einen einfachen Texteditor. Es gibt hier eine große Auswahl, informieren Sie sich im Internet. Für unsere Zwecke brauchbare Editoren sind z.B. `Notepad++` unter Windows, `TextMate` unter MacOS oder `Nano` unter Linux.

Wir wollen nun das "HalloWelt"-Programm aus der Vorlesung übersetzen und ausführen. Gehen Sie dazu wie folgt vor:

a) Erstellen Sie einen Ordner `Prog_1` und darin einen weiteren Ordner namens `Blatt_1` in Ihrem Heimatverzeichnis (verwenden Sie Laufwerk `N:` auf den Poolrechnern), und in diesem wiederum ein Verzeichnis namens `Aufgabe_1`. Wechseln Sie in dieses Verzeichnis und legen dort die Datei `HalloWelt.java` mit einem Texteditor an. Tippen Sie das HalloWelt-Programm von Kapitel 0, Folie 9 in diese Datei und speichern es ab.

b) Sie können das HalloWelt-Java-Programm nicht direkt ausführen, sondern müssen es zuerst in "Java Bytecode", eine Zwischensprache, übersetzen ("compilieren"). Dies bewerkstelligen Sie mit dem Befehl

   ```
   javac HalloWelt.java
   ```

   Dadurch wird eine Datei `HalloWelt.class` erzeugt, die den Bytecode des Programms enthält. Ausgeführt werden kann der Bytecode dann mittels

   ```
   java HalloWelt
   ```

   (D.h., die Dateiendung `.class` muss hier nicht mit angegeben werden.)

c) Schreiben Sie ein vierzeiliges Gedicht (oder suchen Sie eines im Internet). Modifizieren Sie das HalloWelt-Programm so, dass es Ihr Gedicht ausgibt. Ändern Sie dabei auch den Klassennamen von "HalloWelt" in "Gedicht" ab, und speichern Sie Ihr Programm unter `Gedicht.java`. Compilieren Sie Ihr Programm mittels `javac` und führen Sie es mittels `java` aus.

Hinweis: Diese Aufgabe muss von jedem Studierenden, und nicht nur jedem Team, bearbeitet und abgegeben werden. Sie können die Lösungen aller Teammitglieder aber sammeln und gemeinsam abgeben.

Aufgabe 2 (OO-Modellierung): Dr. Bob möchte eine Anwendung programmieren, welche die Erstellung von Grafiken ermöglicht, die aus verschiedenen geometrischen Formen zusammengesetzt sind.

Die unterschiedlichen Formen der Applikation sollen sein: Kreis, Quadrat, Rechteck und Dreieck. Diese sollen auf einer Zeichenfläche mit einer vorher festgelegten Größe angeordnet sein. Jede Form soll eine Farbe besitzen und entweder ausgefüllt sein oder nicht. Formen können mehrfach (mit unterschiedlichen Positionen und Farben) auf der Zeichenfläche vorkommen oder auch gar nicht.

Für einen ersten Test möchte Dr. Bob auf einer Zeichenfläche der Größe 1000x800 (1000 Punkte in x-Richtung, 800 Punkte in y-Richtung) vier Formen platzieren:

1. Einen grünen, ausgefüllten Kreis mit Radius 100 an Position (300,200).

2. Ein rotes, gleichseitiges Dreieck ohne Füllung mit Seitenlänge 200 an Position (600,300).
3. Ein gelbes, ausgefülltes Quadrat mit Seitenlänge 10 an Position (400, 50).
4. Ein blaues, ausgefülltes Rechteck der Größe 600x10 an Position (50, 700).

Eine Position ist dabei gegeben durch ein Paar (x,y), das für die x- und y-Koordinaten der Form steht.

a) Überlegen Sie sich für die Zeichenfläche und die vier genannten Formen der "realen Welt" virtuelle Gegenstücke in der "Programmwelt" und geben Sie diese an. Verwenden Sie dazu für die Objekte der Programmwelt die grafische Notation aus der Vorlesung. Objektnamen können Sie frei wählen, die Methoden-Blöcke bleiben leer, Attribute wählen Sie passend.

   Beachten Sie bitte die Java-Namenskonventionen. Hinweis: Sie erstellen hier *kein* Klassendiagramm, sondern ein *Objektdiagramm*.

b) Die oben angeführte textuelle Beschreibung ("Spezifikation") der Formen und deren Anordnung auf der Zeichenfläche ist nicht eindeutig und erlaubt verschiedene Interpretationen. Wo erkennen Sie Mehrdeutigkeiten? Wie könnten diese aufgelöst werden?

c) Erstellen Sie für die vier möglichen Formen und die Zeichenfläche jeweils eine Java-Klasse. Die Klassen sollen die folgende Form haben:

```
class <name> {
   <type 1> <attribute name 1>; // <comment>
   <type 2> <attribute name 2>; // <comment>
   ...
   <type n> <attribute name n>; // <comment>
}
```

   mit je einer Zeile für jedes relevante Attribut. Für die Typen können Sie verwenden:

   1. `int`, wenn das Attribut ganzzahlige Werte annehmen soll,
   2. `String` für Zeichenketten (Worte),
   3. `float` für Fließkommazahlen ("Kommazahlen"),
   4. `boolean` für Wahrheitswerte (wahr/falsch).

   Speichern Sie alle Klassen in einer einzigen Datei `DrBob.java`.

   Hinweis: Es ist hier nicht geplant, von diesen Klassen Objekte zu erzeugen oder eine lauffähige Konsolenanwendung zu programmieren. Denken Sie sich die hier programmierten Klassen lieber als Datentypen, die anderswo in einem größeren Softwareprojekt verwendet werden könnten.

d) Übersetzen Sie die Datei `DrBob.java` mit dem Java-Compiler `javac`. Falls Fehler gemeldet werden, beheben Sie diese.

e) Welche Dateien werden bei der Compilierung in Teilaufgabe d) erzeugt?

f) **(Freiwillige Zusatzaufgabe für Experten)** Verwenden Sie das Kommando `javap`, um den erzeugten JVM-Code der `class`-Dateien zurück zu übersetzen ("disassemblieren"). Gibt es Unterschiede zu Ihrer Ursprungs-Java-Datei? Experimentieren Sie auch mit den Optionen `-v` und `-c` des `javap`-Kommandos und versuchen Sie, die Ausgaben zu verstehen.

Aufgabe 3 (Datenmodell Freundschaftsbuch):

Hinweis: Im Folgenden bezeichnet jeder **fett** gedruckte Begriff eine Klasse, die Sie modellieren sollen. Begriffe wie *hat* oder *enthält* beziehen sich auf Attribute der Klasse.

Erinnern Sie sich an Freundschaftsbücher, wie Sie zum Beispiel unter Grundschülern gerne benutzt werden? Sehr gut! Hier sollen Sie eines modellieren. Ein **Freundschaftsbuch** *hat* einen Eigentümer und eine feste Zahl an (möglicherweise noch freien) **Einträgen**. Wir entscheiden uns hier der Einfachheit halber für genau **fünf Einträge**. Jeder **Eintrag** *hat* einen **Autor** und *enthält* einen Freitext mit besten Wünschen. Ein **Autor** *hat* einen Namen, ein **Geburtsdatum** und ein Geschlecht. Das **Geburtsdatum** *besteht aus* einem Tag, einem Monat und einem Jahr. Außerdem *hat* ein **Autor** eine Reihe von **Vorlieben**. Die **Vorlieben** *beinhalten* eine Farbe, eine Tierart, ein Buch, einen Film und ein Musikstück.

Erstellen Sie in einer Datei `Friendbook.java` die Klassen `Friendbook`, `FriendbookEntry`, `Author`, `AuthorPreferences` und `Birthdate`. Fügen Sie die oben beschriebenen Attribute an passender Stelle hinzu und überlegen Sie sich geeignete Datentypen, ähnlich wie in Aufgabe 2c). Als Datentypen können Sie auch bereits definierte Klassen verwenden.

Stellen Sie sicher, dass sich Ihre Java-Quellcode-Datei (`Friendbook.java`) compilieren lässt, und beheben Sie gegebenenfalls Fehler.

Hinweis: Sie können (und sollten hier) die Aufgabe mit dem in der Vorlesung bisher gelernten Stoff lösen; insbesondere ist hier nicht vorgesehen, Klassen aus dem Paket `java.util` zu verwenden.