Aufgabenzettel 5
Klassenhierarchien

Dieser Aufgabenzettel muss spätestens in Ihrer Praktikumsgruppe in **KW 22** vorgestellt werden.

Alle für diesen Aufgabenzettel entwickelten Programme sollen sich in einem Paket **de.hsruhrwest.oop.ss2025.classes** oder entsprechenden **Unterpaketen** befinden.

Aufgabe 1: Einfache Klassen

Modellieren Sie Klassen für die folgenden Entitäten:

- Studierende(r)
- Politische Partei, inkl. einer Mitgliederliste
- Nachspeise
- Haustier

Jede Klasse sollte mindestens fünf Attribute besitzen, dazu einen Konstruktor, der alle Attribute setzen kann, Getter, Setter und eine toString-Methode. Achten Sie auf Information Hiding / Datenkapselung.

Aufgabe 2: Klassenhierarchie

Implementieren Sie die folgende Klassenhierarchie. Jede Klasse sollte einen Konstruktor, Getter, Setter und eine toString-Methode besitzen, die eine gut lesbare Form der Objekt-Attribute zurückgibt.

Testet Sie Ihren Code durch ein Hauptprogramm.

a) Schreiben Sie eine Oberklasse für eine **Prüfung**. Eine Prüfung besitzt einen Namen als Attribut. Fügen Sie eine Methode isPassed() („Ist bestanden“) hinzu, die zurückgeben soll, ob eine Prüfung bestanden ist. Unterklassen sollen diese Methode überschreiben. In der

Oberklasse soll die Methode **true** zurückgeben. Der Getter für den Namen der Prüfung soll von einer Unterklasse nicht überschreibbar sein.

b) Implementieren Sie die folgenden Klassen, die von der Oberklasse aus a) **erben**:

- Ein Klasse, die eine **Klausur** modelliert, besitzt Attribute für die Gesamtpunktzahl und die erreichte Punktezahl. Eine Klausur gilt als bestanden (isPassed), wenn mindestens die Hälfte der Punkte erreicht wurde.

- Eine Klasse, die ein **einfaches Praktikum** modelliert, besitzt Attribute für die Anzahl der Praktikumstermine und die Anzahl der besuchten Termine des Praktikums. Ein einfaches Praktikum gilt als bestanden (isPassed), wenn es maximal zwei Fehltermine gibt, in denen ein Studierender nicht anwesend war.

- Eine Klasse, die eine **theoretische Führerscheinprüfung** modelliert, besitzt ein Attribut, das alle gesamten Fehlerpunkte in einem Array speichert. Beispiel wenn eine Person eine 5- und drei 3-Punkte-Fragen falsch beantwortet hat, kann das Array so aussehen: [3, 3, 5, 3]
  Die theoretische Führerscheinprüfung gilt als bestanden, wenn es in Summe maximal 10 Fehlerpunkte gibt, aber keine zwei Fehlerpunkte der Höhe 5.

- Eine Klasse, die das **Aufblasen von Luftballons** als Prüfung modelliert. Die Klasse darf **nicht ableitbar** sein. Es gibt zwei Attribute: eines für das Volumen des aufgeblasenen Luftballons und eines, das modelliert, ob der Luftballon geplatzt ist. Ein geplatzter Ballon muss stets ***null*** als Volumen hinterlegen. Ein intakter Ballon muss ein ganzzahliges Volumen besitzen (nicht *null*). Stellen Sie in der Klasse sicher, dass dies der Fall ist. Die Prüfung ist bestanden, wenn der Luftballon mindestens 2 Liter Luft enthält (und offensichtlich nicht geplatzt ist).

c) Implementieren Sie eine Klasse für eine **Prüfungssammlung**. Sie speichert ein Array von Prüfungen (siehe a und b) für eine Person. Schreiben Sie einen Konstruktor, der die Menge setzt. Fügen Sie die folgenden Getter hinzu:

    - ein Getter gibt die Anzahl der bestandenen Prüfungen zurück.

    - ein Getter gibt die Bestehensquote[1] zurück.

    - ein Getter gibt zurück, wieviele Versuche zum Bestehen der Führerscheinprüfung erforderlich waren. Wurde die Führerscheinprüfung (noch) nicht bestanden, soll 0 zurückgegeben werden. (Tipp: instanceof)

    - ein Getter gibt zurück, wieviele Luftballons bei Prüfungen geplatzt sind.
      (Tipp: instanceof)

[1] Das ist die Anzahl der bestanden Prüfungen dividiert durch die Gesamtzahl aller Prüfungen.