Hochschule Karlsruhe
University of Applied Sciences

Fakultät für **Informatik**
und Wirtschaftsinformatik

Prof. Dr. Carsten Sinz
Lehrgebiet Software-Entwicklung

Übungsblatt 6 zur Vorlesung Programmieren (fällig bis 14.12.25)

Aufgabe 1 (Objektorientierte Modellierung Fahrrad):

Ein Fahrradhändler möchte ein Programm für die Verwaltung seiner Fahrräder und deren Komponenten entwickeln, und Sie sollen ihm dabei helfen. Er verkauft Fahrräder verschiedener Kategorien:

- Rennräder,
- Mountainbikes (MTBs) und
- Stadtfahrräder.

Bei den Mountainbikes und den Stadtfahrrädern gibt es jeweils auch E-Bike-Ausführungen. Jedes Fahrrad ist aus verschiedenen Komponenten zusammengesetzt. Die hier betrachteten sollen sein:

- Rahmen (Carbon oder Alu)
- Bremsen (Felgen-, Scheiben- oder Trommelbremsen)
- Schaltung (Kettenschaltung oder Nabenschaltung)
- Reifen

Von jedem Komponententyp gibt es verschiedene Ausführungen, die alle (zumindest) einen Namen (z.B. Markenname wie "Shimano Deore LX") besitzen sollen. Das Namens-Attribut ist dagegen *nicht* für den Datentyp-Namen (z.B. `DiscBrake`) vorgesehen.

Die verschiedenen Ausführungen der Komponententypen sollen eigenständige Datentypen sein, keine `enum`-Konstanten.

Ihre Aufgabe ist nun wie folgt:

a) Entwerfen Sie die Klassenhierarchien und zeichnen diese wie in der Vorlesung als Baum auf. Notieren Sie auch, welche der Klassen abstrakt sein sollen.
   - ... für Fahrräder; die Basisklasse soll dabei `Bike` heißen.
   - ... für die Fahrrad-Komponenten Rahmen, Bremse und Schaltung.
   - Überlegen Sie sich Vor- und Nachteile einer hypothetischen gemeinsamen Basisklasse `BikeComponent` für die Fahrrad-Komponenten.

b) Legen Sie Java-Klassen an für sämtliche in a) entworfenen Klassen:
   - Fügen Sie den Fahrradklassen Attribute für die Komponenten hinzu.
   - Fahrräder sollen benannt werden können. Fügen Sie dazu an passender Stelle ein Attribut `String name` hinzu sowie `get-` und `set`-Methoden für dieses Attribut. Das `name`-Attribut soll dabei über die Klassenhierarchie vererbt werden. Fügen Sie der Klasse Rahmen außerdem noch ein Attribut für eine Farbe hinzu.

     b.w.

- Nachdem ein Fahrrad gebaut wurde, sollen seine Komponenten nicht mehr änderbar sein.
- Schreiben Sie `toString()`-Methoden überall wo nötig – zumindest also in Ihren allgemeinsten Basisklassen. Beachten Sie hierzu auch die Hinweise zu dieser Methode in der offiziellen Dokumentation zur Klasse `Object`.
- Insbesondere sollte die `toString()`-Methode der Klasse `Bike` eine vollständige Beschreibung des Fahrrads liefern: Namen und Typ des Fahrrads, und welche Komponenten (Name und Typ) verbaut sind.

c) Erstellen Sie in der `main`-Methode einer weiteren Klasse Objekte für die drei folgenden Fahrräder (Namen können Sie frei wählen) und geben Sie deren Beschreibung (Name und Komponenten) aus:

1. Rotes Rennrad mit Carbon-Rahmen, Felgenbremsen und Kettenschaltung
2. Schwarzes E-Mountainbike mit Alurahmen, Scheibenbremsen und Kettenschaltung
3. Weißes E-Stadtfahrrad mit Alurahmen, Trommelbremsen und Nabenschaltung.

Tipp: Wenn für einen Objekttyp eine `toString()`-Methode programmiert ist, wird diese automatisch aufgerufen, falls Sie ein Objekt an die Methode `System.out.println()` übergeben.

Aufgabe 2 (Zahlenklassen in Java):

In dieser Aufgabe wollen wir uns mit den in der Java-Standard-Bibliothek vorhandenen Klassen für Zahlen vertraut machen. Bearbeiten Sie dazu die folgenden Teilaufgaben:

a) Lesen Sie sich die Dokumentation zur Java-Klasse `Number` des Pakets `java.lang` durch.
- Wieso sind manche Methoden dieser Klasse als abstrakte Methoden deklariert, andere hingegen nicht?
- Geben Sie einen Grund an, warum das Vorhandensein der Superklasse `Number` generell sinnvoll sein könnte.

b) Betrachten Sie die Klasse `Integer`.
- Welche statischen Attribute besitzt diese Klasse?
- Schreiben Sie ein kleines Programm, das die Werte dieser Attribute ausgibt.
- Wozu dienen die Methoden `bitCount`, `rotateLeft`, `numberOfTrailingZeros` und `reverse`? Können Sie sich Anwendungen für diese Methoden vorstellen?
- Ergänzen Sie Ihr Programm so, dass es die Ergebniswerte dieser vier Methoden ausgibt angewandt auf die Eingabedaten 100 sowie -100. (Bei `rotateLeft` dürfen Sie den zweiten Parameter beliebig wählen.)

c) Betrachten Sie die Klassen `BigInteger` und `BigDecimal`.
- Welche Zahlenwerte lassen sich mittels dieser beiden Klassen repräsentieren?
- Wodurch unterscheiden sich die Klassen?

b.w.

d) Schreiben Sie ein Programm, das die Fakultätsfunktion $n!$ auch für große Eingaben n berechnen kann. Verwenden Sie für die Zahlendarstellung die Klasse `BigInteger`, für den Parameter n jedoch `int` (warum reicht das aus?).

Die Fakultätsfunktion ist definiert durch:

$$n! \quad = \quad \begin{cases} \prod_{i=1}^{n} i \quad = \quad n \cdot (n-1) \cdot \ldots \cdot 1 & , \quad n > 0 \\ \\ 1 & , \quad n = 0 \end{cases}$$

Die Zahl n soll über die Konsole durch den Benutzer eingegeben werden können.