Übung 1: Klassendiagramm

Erstellen Sie aus den folgenden Aussagen die entsprechenden **Klassendiagramme**! Diskutieren Sie mit Ihren Sitznachbarn. Beachten Sie die bereitgestellte UML Notations-Übersicht.

- Studenten haben einen Namen und eine Matrikelnummer sowie Prüfungsergebnisse. Studenten werden einem Studiengang zugeordnet.
- Hersteller produzieren Artikel. Ein Artikel hat eine eindeutige Artikelnummer und eine Bezeichnung. Handelt es sich um ein Lebensmittel, hat der Artikel zudem noch ein Mindesthaltbarkeitsdatum.
- Ein Fragebogen besteht aus Freitextfragen, Einfachselektionen und Mehrfachselektionen. Auch Bilder und formatierten Text kann ein Fragebogen enthalten; diese werden aber vom Befragten nicht beantwortet.
- Ein Stack ist eine spezielle Liste.
- Eine Autovermietung vermietet Fahrzeuge. Die Fahrzeuge haben eine spezielle Ausstattung. Kunden können Fahrzeuge an einem Standort anmieten und an einem beliebigen anderen Standort wieder zurückgeben. Bei Rückgabe werden mögliche Schäden und der Benzinstand protokolliert.
- Versicherungsnehmer können natürliche oder juristische Personen sein. Ein Versicherungsnehmer hat mehrere Verträge. Ein Vertrag hat eine eindeutige Vertragsnummer und eine Laufzeit. Jeder Versicherungsnehmer besitzt genau eine Adresse.
- Ein Automobilhersteller stellt verschiedene Autos her. Jedes Auto hat vier Räder. Jedes Rad gibt es in verschiedenen Größen und mit verschiedenen Felgen. Ohne Räder ist es kein Auto.
- Herr Müller vermietet eine Wohnung in der Lindenstr. 33 an Frau Huber für 400 EUR im Monat. Der gemeinsame Mietvertrag hat eine Laufzeit von 5 Jahren.

Übung 2: Klassenhierarchien

Gestern haben wir die Klasse `RacingCar` implementiert. Heute bringen wir etwas mehr Struktur in unseren Klassenentwurf. In dieser Aufgabe entwerfen und implementieren Sie eine Klasse `Vehicle`, welche als generelle Oberklasse für andere konkrete Klassen dienen soll. Die Klasse `Vehicle` hat folgende Attribute:

- `modelName` speichert die Bezeichnung des Fahrzeugmodells.
- `maxSpeed` speichert die maximale Geschwindigkeit des Fahrzeugs und soll über eine Methode `setSpeed` gesetzt werden können.
- `brand` speichert den Hersteller des Fahrzeugs und soll als eigene Klasse `Manufacturer` modelliert werden.

Die Klasse `Vehicle` hat außerdem eine Methode `printInfo()`, welche die Details des Fahrzeugs auf der Konsole ausgibt.

Die Klasse `Manufacturer` hat nur ein Attribut `name`, das den Namen des Herstellers speichert.

Nun wollen wir zwei konkrete Klassen `Car` und `Bicycle` hinzufügen, die von `Vehicle` abgeleitet werden.

Die Klasse `Car` hat zwei weitere spezielle Attribute:

- `doors` speichert die Anzahl Türen.
- `power` gibt an, wie viele PS das Auto hat.

Die Klasse `Bicycle` hat ein weiteres spezielles Attribut:

- `gears` speichert die Anzahl Gänge in der Gangschaltung des Fahrrads.

Nun Ihre Aufgaben:

1. Implementieren Sie die oben erklärte Klassenstruktur. Verwenden Sie die bereitgestellten Templates und vervollständigen Sie diese!
2. Was fällt Ihnen bei der Vererbung der Konstruktoren auf?
3. Machen alle Attribute so Sinn in dieser Vererbungsbeziehung?
4. Beide Klassen `Car` und `Bicycle` sollen ein Attribut `isElectric` bekommen, das angibt, ob ein Elektromotor verbaut ist. Was gibt es bei der Implementierung zu beachten?

Übung 3: Fahrzeugverwaltung

In der vorherigen Aufgabe haben wir den Grundstock der Klassenhierarchie für eine Fahrzeugverwaltung gelegt. Ähnlich wie gestern implementieren wir nun eine menügesteuerte Benutzerführung. Die Fahrzeugverwaltung soll folgende Funktionen unterstützen:

- *anzeigen* aller bestehenden Fahrzeuge
- *anlegen* von neuen Fahrzeugen
- *löschen* von bestehenden Fahrzeugen

Beachten Sie, dass dabei nicht im Vordergrund steht, optische/ergonomische Aspekte der Benutzeroberfläche zu optimieren. Sondern es soll darum gehen, dass Sie ein besseres Gefühl für die Gesamtzusammenhänge der Objektorientierung und der Programmiersprache Java selbst bekommen.

Ihre Aufgaben:

1. Implementieren Sie zuerst die oben genannten Funktionen anhand der bereitgestellten Templates und Ihrer Klassenhierarchie aus der vorherigen Aufgabe.
2. Nachdem Sie alle Funktionen implementiert haben, was fällt Ihnen an der Struktur der Klasse `VehicleManagement` auf? Könnte man es besser strukturieren?
3. (Bonus) Erweitern Sie die Funktionalität, sodass Fahrzeuge auch *ausgeliehen* werden können. Wie müssen Sie das Menü und die Klassenhierarchie entsprechend erweitern? Achtung: Natürlich sollte man auch kein Fahrzeug löschen oder ausleihen können, wenn es schon ausgeliehen ist. Gibt es noch weitere Funktionen, die dabei ergänzt werden müssen?
4. (Bonus) Das Anzeigen aller bestehenden Fahrzeuge soll nun *sortiert* ausgegeben werden. Wie können Sie das realisieren?