Kieback&Peter

Java: Objektorientierte Programmierung 4 / UML + Vererbung

Aufgabe 1

Recherchiert welche UML Tools es gibt. Im besten Fall sollten Klassendiagramme mit den meisten Notationen abgebildet werden können. Dabei sollte es ich um Open Source Programme oder kostenlose Web Versionen handeln.

Für die Nachfolgenden Aufgaben sollt ihr mit diesem Tool die Klassendiagramme nach Aufgabenstellung erstellen.

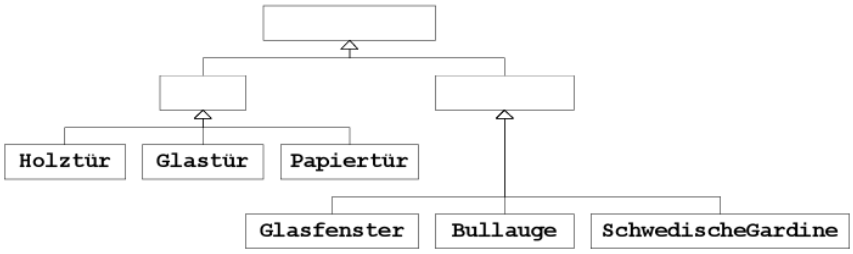
Aufgabe 2

Bringt folgende Klassen in eine Sinnvolle Hierarchie, indem ihr ein Klassendiagramm mit einer Generalisierung entwerft.

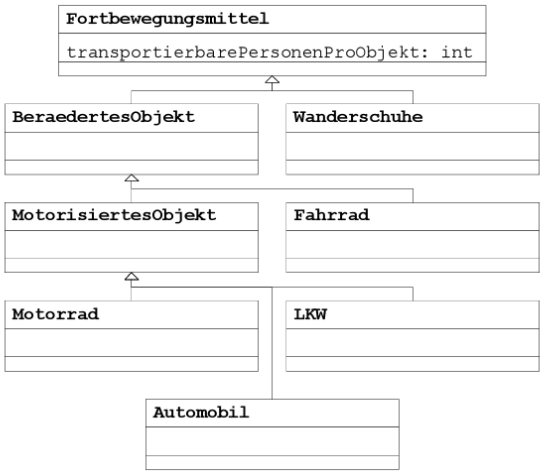
Tier, Wirbeltier, Lebewesen, Insekt, Pflanze, Fisch, Biene, Säugetier, Katze, Hund, Delphin, Maus, Venusfliegenfalle, Aal, Hauskatze, Hamster

Aufgabe 3

Füllt die Lücken im Klassendiagramm



Aufgabe 4



In die oberste Klasse Fortbewegungsmittel haben wir eine Instanzvariable namens transportierbarePersonenProObjekt eingetragen, die mittels eines intWertes die maximale Zahl von Personen darstellt, die in einem speziellen Fahrzeugobjekt transportiert werden können.

Da sich alle anderen Klassen von der Superklasse ableiten, erben sie diese Variable automatisch, sodass wir sie nicht in jeder Klasse erneut definieren müssen. Tragt in diesem Sinne die drei folgenden Instanzvariablen an der richtigen Stelle ins UML-Diagramm ein: eine Variable maximaleGeschwindigkeit vom Typ int , die die maximale Geschwindigkeit in Metern pro Sekunde codiert, die man mit diesem Fortbewegungsmittel erreichen kann, eine Variable zahlDerRaeder vom Typ int und eine double-Zahl namens motorLeistungInPS , die die Leistung eines Motors codiert.

Aufgabe 5

Erklärt den Unterschied zwischen einer Abstrakten Klasse und einer Schnittstelle in Java. Am besten findet ihr ein aussagekräftiges Minimalbeispiel dazu.

Aufgabe 6

Leitet von der Klasse Konto zwei Klassen - GiroKonto und SparKonto - ab.

GiroKonto erhält zusätzlich ein Überziehungslimit und einen Soll-Zinssatz.

SparKonto erhält nur einen Haben-Zinssatz

Definiert beide Klassen mit Konstruktoren, mit Zugriffsmethoden zur Änderung aller Werte und mit je einer display-Methode.

Aufgabe 6.1

Erstellt ein Klassendiagramm

Aufgabe 6.2

Java Implementierung:

Legt die Konto-Klasse in das Paket *de.kiebackpeter.konto*
und die anderen Klassen in das Paket *de.kiebackpeter.konto.spezial.*

Testet die neuen Klassen mit einer Klasse KontoTest, die ihr im Default-Paket anlegt.

Ablauf des Tests:
Definition und gleichzeitige Initialisierung eines Kontos vom jeweiligen Typ mit anschließender Ausgabe der Kontodaten. Darauf Änderung beider Konten mit erneuter Ausgabe der Kontodaten.

Aufgabe 6.3

Verändert die Kontoklasse.

Überlagert die Methode equals, so dass sie für verschiedene Kontoobjekte, die gleiche Inhalte haben,
korrekt arbeitet.

...

Führt die bisherige display-Methode in die toString() Methode über und testen Sie das Verhalten, in dem ihr Kontoobjekte mit System.out.println ausgeben.

Aufgabe 7

Realisiert eine Klassenhierarchie, die sich aus den Klassen Form, Form1dimensional, Form2dimensional, Form3dimensional, Dreieck, Rechteck, Kreis, Würfel, Zylinder und Linie zusammensetzt.

Implementiert die Methoden berechneFläche() , berechneUmfang() und berechneVolumen() an den korrekten Stellen dieser Klassenhierarchie.

Aufgabe 7.1

Erstellt ein Klassendiagramm

Aufgabe 7.2

Implementierung in Java

Aufgabe 8

Für die Simulation eines Wettrennens sollen verschiedene Fahrzeugarten objektorientiert modelliert werden. Da alle Fahrzeugtypen gemeinsame Eigenschaften haben, definieren wir uns zunächst eine Basisklasse Fahrzeug, die als Oberklasse für die anderen Klassen dienen soll.

Modelliert dazu zuerst ein Klassendiagramm und implementiert dieses dann in Java.

Ein Fahrzeug hat folgende allgemeinen Merkmale:

- Seine aktuelle Position (in km und der Einfachheit halber in nur einer Dimension)
- Seine aktuelle Geschwindigkeit (in km/h)
- Es kann bewegt werden (Methode bewege). Die Methode wird mit einem double-Parameter aufgerufen, der die Anzahl der Minuten angibt, die sich das Fahrzeug mit der aktuellen Geschwindigkeit vorwärtsbewegt. Der Methodenaufruf ändert die Position des Fahrzeugs, wenn es mit einer von 0 verschiedenen Geschwindigkeit bewegt wird.
- Man kann seine Geschwindigkeit setzen (Methode setzeGeschwindigkeit). Die Geschwindigkeit darf die Maximalgeschwindigkeit nicht überschreiten, eine korrekte Ausführung sollte protokolliert werden.
- Es kann seine Maximalgeschwindigkeit angeben (Methode getMaxGeschwindigkeit). Für ein Objekt der Klasse Fahrzeug soll die Maximalgeschwindigkeit 0 sein.
  Es kann die Anzahl seiner Räder angeben. In der Klasse Fahrzeug soll diese ebenfalls 0 sein.

...

Nun sollen einige konkrete Fahrzeuge definiert werden, indem entsprechende Klassen von Fahrzeug abgeleitet werden:

- Ein Fahrrad ist ein Fahrzeug mit 2 Rädern und Maximalgeschwindigkeit 30 km/h.
- Ein Auto ist ein Fahrzeug mit 4 Rädern und Maximalgeschwindigkeit 140 km/h.
- Ein Rennwagen ist ein Auto mit Maximalgeschwindigkeit 220 km/h.
- Ein Krankenwagen ist ein Auto mit einem zusätzlichen Blaulicht, das ein- oder ausgeschaltet sein kann (neues Attribut!). Außerdem muss der Krankenwagen Methoden zum Ein- bzw. Ausschalten des Blaulichts anbieten.

Definiert diese Klassen und nutzt dabei so weit wie möglich die Vererbung von Eigenschaften aus! Erstellt dazu ebenfalls ein Objektdiagramm (Fahrrad, Krankenwagen).

Nun soll die eigentliche Simulation des Wettrennens in einer Klasse Wettrennen geschrieben werden.

Erzeugt je ein Fahrzeug und setzen Sie dann die Geschwindigkeiten auf:

- Fahrrad 20 km/h
- Auto 150 km/h
- Rennwagen 200 km/h
- Krankenwagen 80 km/h

Dann sollen sich die Fahrzeuge bewegen. Der Gerechtigkeit halber geben wir dem Fahrrad einen Vorsprung von 4 Stunden. Danach lasst alle Fahrzeuge eine Stunde lang mit unveränderter Geschwindigkeit vorwärtsfahren. Was fällt Ihnen beim Auto auf?

Aufgabe 9

Wie kann eine Mehrfachvererbung in Java realisiert werden? Zeigt dieses an einem Minimalbeispiel.

Hinweis:
Erstellt für die Bearbeitung der Aufgaben ein Git Repository und legt die Aufgaben daran ab.