Programmieren | OOP

Style Guide | Konventionen

Klasse

Eine Klasse beschreibt eine **Menge von Objekten mit gemeinsamer Semantik, gemeinsamen Eigenschaften und gemeinsamen Verhalten**. Die Eigenschaften werden durch Attribute und das Verhalten durch Operationen abgebildet. Klassen besitzen – mit Ausnahme von abstrakten Klassen – einen Mechanismus, um neue Objekte zu erzeugen.

`public [abstract] class CarPark`	Der **Name** einer Klasse **startet mit** mindestens einem **Großbuchstaben** und wird ggf. in CamelCase-Notation geschrieben.

Attribute

Attribute repräsentieren die strukturellen Eigenschaften von Klassen. Sie bilden somit das **Datengerüst** von Klassen. Alle Objekte einer Klasse besitzen dieselben Attribute, jedoch im Allgemeinen unterschiedliche Attributwerte. Jedes Attribut ist von einem bestimmten Typ und kann einen Anfangswert (default) besitzen.

`boolean isSelfDriving`	Der **Name** eines Attributs **startet mit** mindestens einem **Kleinbuchstaben** und wird ggf. in camelCase-Notation geschrieben.

Datentypen | Enumeration

Aufzählungstypen sind spezielle Datentypen mit einer **endlichen Menge benutzerdefinierter, diskreter Werte**, Ein solcher Wert heißt Aufzählungswert (EnumerationLiteral) und spezifiziert einen Wert, den ein Element mit dem Aufzählungstyp zur Laufzeit annehmen darf.

`public enum Signal {`
`  GREEN, RED`
`}`
	Der **Aufzählungswert** wird **in GROSSBUCHSTABEN** geschrieben.

Datenstrukturen | Array und ArrayList

Ein Array (Feld) speichert eine **Folgen von Werten**, die alle den **gleichen Datentyp** haben.

Datenstrukturen | Stack

Ein Stapel (oder auch Kellerspeicher oder Stack) ist eine **Collection**, die auf dem **LIFO-Prinzip** (Last-in-First-out) basiert.

Datenstrukturen | Queue

Eine FIFO-Warteschlange (queue) ist eine Collection, die auf der First-in-First-out-Strategie (**FIFO**) basiert.

Datenstrukturen | HashMap

Die Hashmap ist eine **Zuordnung, die jedem Schlüssel einen Datensatz zuordnet**. Ziel ist es die Hashmap so aufzubauen, dass über den Schlüssel schnell auf den zugehörigen Datensatz zugegriffen werden kann.

Operationen

Operationen werden eingesetzt, um das **Verhalten von Objekten** zu beschreiben. Alle Operationen einer Klasse zusammengenommen, definieren die Möglichkeiten zur Interaktion mit Objekten dieser Klasse. Eine Operation kann **Botschaften** an andere Objekte **senden**. Gleichzeitig stellen Operationen die einzige Möglichkeit dar, Zustandsänderungen eines Objektes herbeizuführen.

`public int[][] scanFace() {...`	Der **Name** eines Attributs **startet mit** mindestens einem **Kleinbuchstaben** und wird ggf. in camelCase-Notation geschrieben.

Schnittstelle | Interface

Eine Schnittstelle (interface) beschreibt eine **Menge von öffentlichen Operationen**, Merkmalen und „Verpflichtungen“, die durch einen Classifier, der die Schnittstelle implementiert, zwingend bereitgestellt werden müssen. Schnittstellen werden nicht instantiiert, sondern durch einen Classifier realisiert.

`public interface ICamera {...`	Der **Name** einer Schnittstelle **startet mit** dem **Präfix I**, **mindestens einem folgenden Großbuchstaben** und wird ggf. in CamelCase-Notation geschrieben.

Assoziation

Eine Assoziation beschreibt eine **Menge gleichartiger Beziehungen zwischen Klassen**. Dabei kann die Beziehung als Weg oder Kommunikationskanal verstanden werden. Die Assoziation definiert eine sehr enge Form der Beziehung zwischen Klassen, die das gegenseitige Zugreifen auf Elemente der Klasse (Attribute und Operationen) ermöglicht. Assoziationen bauen somit ein Netzwerk zwischen losen, vereinzelten Klassen auf.

Vererbung | Generalisierung

Die Generalisierung beschreibt die **Beziehung zwischen einer allgemeinen Klasse (Basisklasse) und einer spezialisierten Klasse**. Die spezialisierte Klasse erweitert die Liste der Attribute, Operationen und Assoziationen der Basisklasse. Operationen der Basisklasse können redefiniert werden. Es entsteht eine Klassenhierarchie im Sinne einer **taxonomische**n **Beziehung**.

Überschreiben

Von Überschreiben bzw. Redefinition spricht man, wenn eine **Unterklasse eine geerbte Operation der Oberklasse – unter dem gleichen Namen – neu implementiert**. Beim Überschreiben müssen die Anzahl und Typen der Ein-/Ausgabeparameter gleich bleiben. Beim Überschreiben muss die Operation der Unterklasse kompatibel mit derjenigen der Oberklasse sein.

Überladen

Überladen bedeutet, dass **mindestens zwei Methoden mit dem gleichen Namen, aber unterschiedlichen Parametern in einer Klasse** deklariert werden können. Der Compiler erkennt dann beim Aufruf der Methode anhand der Anzahl und Typisierung der Parameter, welche Methode gemeint ist.

Paket

Ein Paket fasst mehrere paketierbare Elemente zu einen größeren Einheit zusammen. Ein Paket kann selbst Pakete enthalten. Das Konzept des Pakets wird benötigt, um die **Elemente des Modells** in sinnvoller zu **gruppieren** und die Systemstruktur auf einer hohen Abstraktionsebene zu beschreiben. Durch ein Paket wird ein Namensraum definiert, dem alle Mitglieder des Pakets angehören.

`package facedetectionsystem`
 	
`Nicht: ~~face_detection_system~~`
	Der **Name** eines Pakets **wird fortlaufend mit Kleinbuchstaben** ohne _ usw. geschrieben.

Für die Bezeichnung der Klassen, Attribute und Methoden sind englische Bezeichnungen (BE) zu verwenden.

Die Hinweise der Code Inspection in IntelliJ sind weitestgehend zu berücksichtigen.

Der Source ist um unnötige Leerzeilen, nicht verwandte Attribute und Methoden zu bereinigen sowie in IntelliJ mit Code ► Reformat zu formatieren.

Die Konventionen sind verbindlich.