Klasse (OOP)

Wechseln zu:Navigation, Suche

Dieser Artikel erfüllt die GlossarWiki-Qualitätsanforderungen **nur teilweise**:

Korrektheit: 4 (großteils überprüft)	**Umfang**: 5 (wesentliche Fakten vorhanden)	**Quellenangaben**: 2 (wichtige Quellen fehlen)	**Quellenarten**: 3 (gut)	**Konformität**: 4 (sehr gut)

Inhaltsverzeichnis

1 Definitionen (OMG: UML 2.3)

1.1 Classifier[1]

A **classifier** is a classification of instances — it describes a set of instances that have features in common.

Übersetzung (von W. Kowarschick):

Ein Klassifikator [Classifier] ist eine Klassifikation von Objekten — er beschreibt eine Menge von Objekten, die Eigenschaften gemein haben.

1.2 Class[1]

A class describes a set of objects that share the same specifications of features, constraints, and semantics.

Übersetzung (von W. Kowarschick):

Eine Klasse beschreibt eine Menge von Objekten, die sich dieselben Spezifikationen von Eigenschaften, [Integritäts-]Bedingungen und Semantik teilen.

1.3 Abstract Classifier, Abstract Class[1]

Laut UML-2.3-Standard hat jeder Klassifikator und damit auch jede Klasse ein Attribut `isAbstract`. Für dieses gilt folgende Bedingung:

If true, the Classifier does not provide a complete declaration and can typically not be instantiated.

Übersetzung (von W. Kowarschick):

Falls [dieses Attribut den Wert] wahr [hat], bietet der Klassifikator keine vollständige Deklaration und kann typischerweise nicht instanziiert werden.

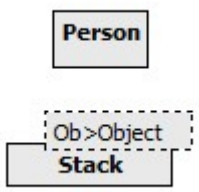
1.4 Interface[2]

An **interface** is a kind of classifier that represents a declaration of a set of coherent public features and obligations. An interface specifies a contract; any instance of a classifier that realizes the interface must fulfill that contract.

Übersetzung (von W. Kowarschick):
Eine Schnittstelle [ein Interface] ist eine [spezielle] Art eines Klassifikators [Classifiers], der die Deklaration einer Menge von kohärenten offentlichen Eigenschaften und Verpflichtungen [im Sinne von Constraints] definiert. Eine Schnittstelle spezifiziert einen Vertrag; jedes Objekt eines Klassifikators, das die Schnittstelle realisiert, muss diesen Vertrag erfüllen.

2 Definitionen (von W. Kowarschick)[3]

2.1 Klasse



UML-Klassen-diagramm: Eine Klasse und ein

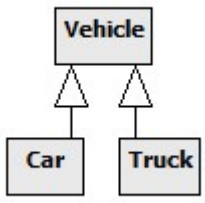
Klassen-Template

Eine **Klasse** dient (im Sinne des objektorienten Paradigmas) dazu, die Schnittstellen und evtl. auch eine oder gar mehrere Implementierungen von „gleichartigen“ Objekten ganz oder zumindest teilweise festzulegen.

Eine **Klasse** besteht aus folgenden Teilen:

einer Klassenextension: Die Klassenextension ist eine (i. Allg. endliche) Menge, die zu jedem Zeitpunkt genau diejenigen Objekte, die der Klasse zugeordnet sind, enthält.
einem Klassenschema (einer Klassenspezifikation): Das Klassenschema umfasst eine (endliche) Menge von Integritätsbedingungen, die alle zugehörigen Objekte (d.h. alle in der Klassenextension enthaltenen Elemente) zu jedem Zeitpunkt erfüllen müssen. Das Schema deklariert insbesondere die Methoden, die für jedes Objekt der Klassenextension jederzeit als Kommunikationsschnittstelle zur Verfügungstehen müssen.
beliebig vielen (0, 1, 2, ...) Klassenimplementierungen: Diese implementieren die im Klassenschema deklarierten Methoden teilweise oder vollständig und garantieren dabei die stete Erfüllung aller im Klassenschema definierten Integritätsbedingungen. Unterschiedliche Implementierungen können sich in Hinblick auf Performanz-Aspekte unterscheiden.
den Klassenmethoden (statische Methoden): Eine Klasse verhält sich wie ein spezielles Objekt) und kann daher eigene (Klassen-)Methoden und (Klassen-)Attribute enthalten. Überlicherweise gibt es zumindest Methoden zur Verwaltung der Klassenextension, d.h. zum Erzeugen (Konstruktoren) und evtl. auch zum Zerstören (Destruktoren) von zugeordneten Objekten.

2.2 Subklasse



UML-Klassen-diagramm: Eine Superklasse und zwei Subklassen

Eine Klasse B heißt **Subklasse** der Klasse A, wenn die Extension von B eine Teilmenge der Extension von A ist und wenn die Integritätsbedingungen der Klasse A logisch aus den Integritätsbedingungen der Klasse B folgen:

Integritätsbedingungen(B) ⇒ Integritätsbedingungen(A)

Das heißt, jedes Objekt aus der Extension von B liegt nicht nur auch in der Extension von A, sondern erfüllt auch alle Integritätsbedingungen von A.

Die Subklasse B erbt alle Implementierungen der Superklasse A, kann diese jedoch „überschreiben“ (ersetzen).

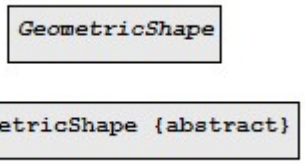
Siehe auch: Vererbung

2.3 Superklasse

Eine Klasse A heißt **Superklasse** der Klasse B genau dann wenn die Klasse B **Subklasse** der Klasse A ist.

Siehe auch: Vererbung

2.4 Abstrakte Klasse

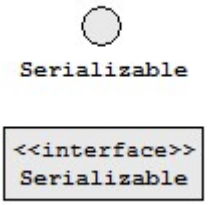


UML-Klassendiagramm: Abstrakte Klasse

Eine Klasse, für die es keinen Konstruktor gibt, der direkt (und nicht nur vom Konstruktor einer Subklasse aus) aufgrufen wernden kann,

heisst **abstrakte Klasse**. Die Klassenimplementierungen, die einer abstrakten Klasse direkt (d.h. nicht indirekt in Subklassen) zugeordnet sind, können unvollständig sein.

2.5 Schnittstelle (Interface)



UML-Klassen-diagramm: Interface

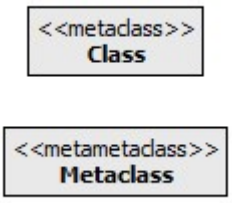
Eine Klasse, für die es keine Klassenimplementierung gibt, heißt **Schnittstelle** oder **Interface**.

2.5.1 Anmerkung

Jede **Schnittstelle** ist auch eine abstrakte Klasse.

Interfaces unterstützen i. Allg. auch keine Klassenmethoden. Dies wurde aber in der obigen Definition nicht explizit gefordert.

2.6 Metaklasse



UML-Klassen-diagramm: Eine Metaklasse und eine Metametaklasse

Eine Klasse, deren Extension zu jedem Zeitpunkt ausschließlich Klassen(objekte) enthält, heißt **Metaklasse**.

2.7 Metametaklasse

Eine Klasse, deren Extension zu jedem Zeitpunkt ausschließlich Metaklassen(objekte) enthält, heißt **Metametaklasse**.

2.8 Metametametaklasse

Und so weiter ...

2.9 Singleton-Klasse

UML-Klassen-diagramm: Singleton-Klasse

Eine Klasse, deren Extension genau ein Objekt enthält, heißt **Singleton-Klasse** oder kurz **Singleton**.

3 Bemerkungen

In der Definition des UML-Standards ist die Klasse ein spezieller Klassifikator. Ein Klassifikator beschreibt eine Menge von Objekten (Instanzen), die gemeinsame Eigenschaften haben. Hier klingen die Begriffe „Klassenextension“ („set of instances“) und „Integritätsbedingungen“ („features“) bereits

an, sind aber recht vage gehalten. Bei der Definition des Begriffes „Klasse“ wird die Extension abermals explizit erwähnt („set of objects“), daruber hinaus wird auch explizit von „constraints“ gesprochen. Methoden und Attribute („features“ einschließlich der zugehörigen „semantics“) werden jedoch gesondert genannt, während sie in der Definition von Kowarschick als spezielle Integritätsbedingungen verstanden werden. Die Methodenimplementierungen sowie die Klassenmethoden fehlen in der UML-Definition vollkommen. Daher unterscheiden sich die beiden Definitionen von „Classifier“ und „Class“ inhaltlich kaum, obwohl das zweite eine Spezialfall des ersten ist.

Schnittstellen werden in UML 2.3 als spezielle Klassifikatoren definiert. Kowarschick definiert sie dagegen als spezielle Klassen. Das heißt, die Definition von UML 2.3 ist etwas allgemeiner (aber auch etwas schwammiger).

Abstakte Klassen werden sowohl in UML 2.3, als auch bei Kowarschick als spezielle Klassen aufgefasst. Allerdings ist die Definition von UML 2.3 missverständlich: Selbstverständlich enthält auch die Klassenextension einer abstrakten Klasse Instanzen. Diese Extension kann allerdings keine Objekte enthalten, die nicht auch Element der Extension einer nicht-abstrakten Subklasse sind. Das heißt, die Extension einer abstrakten Klasse kann keine „eigenen“ Instanzen enthalten.

Die von Kowarschick verwendenten Begriffe Klassenschema und Klassenextension haben ihren Ursprung in der Datenbankwelt: Diese Bezeichnungen betonen die Verwandschaft zwischen objektorientierten Systemen und Datenbanksystemen:

Objektorientierte Systeme	**Datenbanksysteme**
Klasse	Relation/Tabelle
Klassenschema	Relationenschema
Klassenextension	Extension einer Relation (= Menge aller Tupel)
Objekt	Tupel/Entität
Attribut/Zustandsvariable	Attribut

Ja, sogar die in objektorientierten Sprachen so weit verbreitete Punktnotation (`myCar.maxSpeed`) wurde aus der Datenbankwelt (SQL) übernommen.

4 Quellen

1. Unified Modeling Language (OMG UML), Infrastructure, Version 2.3, May 2010
2. [http://www.omg.org/spec/UML/2.3/Superstructure/PDF/ OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.3, May 2010]
3. Kowarschick, W.: Multimedia-Programmierung

1. **Kowarschick (2002a)**: Wolfgang Kowarschick; Multimedia-Programmierung – Objektorientierte Grundlagen; Hrsg.: Michael Lutz und Christian Märtin; Reihe: Informatik interaktiv; Verlag: Fachbuchverlag Leipzig im Carl Hanser Verlag; ISBN: 3446217002; 2002; Quellengüte: 5 (Buch)

5 Siehe auch

1. Wikipedia: Klasse (objektorientierte Programmierung)

Glossar
Kapitel:Multimedia-Programmierung