Anwendungsentwicklung und Programmierung	Klasse:	Datum:	N°.:1

Die Symbolik der Unified Modelling Language (1)

Die Unified Modelling Language (UML) als Modellierungssprache

Bei der heutigen Softwareentwicklung komplexer Systeme ist die genaue Planung von größter Bedeutung. Der Kunde muss die Vorhaben des Entwicklungsteams verstehen, um Änderungswünsche anzeigen zu können. Die Softwareentwicklung im Team bedingt ebenfalls eine standardisierte Kommunikation, da jedes Teammitglied seine Aufgabe im Gesamtbild der Entwicklung verstehen muss. Mit Hilfe der UML können Systementwickler nun Pläne erstellen, die ihre Visionen in einer standardisierten, leicht verständlichen Form aufnehmen und diese anderen vermitteln.
Erfinder der UML sind Grady Booch, James Rumbaugh und Ivar Jacobsen. Diese "drei Amigos" entwickelten in den neunziger Jahren in unterschiedlichen Unternehmen eigenständige Konzepte und Modelle zum objektorientierten Design. Mitte der neunziger Jahre traten Rumbaugh und Jacobsen der Firma Rational Software bei, in der Grady Booch bereits beschäftigt war. Gemeinsam entwickelten Sie die UML und übergaben sie einem Konsortium (DEC, HP, Microsoft, Oracle u.a.) der Object Management Group (OMG). Diese erstellte 1997 die Version 1.0 der UML. Aufgrund des überragenden Echos der Industrie expandierte die OMG und produzierte 1998 zwei weitere Versionen. Auf diesem Weg wurde die UML ein de - facto - Standard in der Softwareindustrie und entwickelt sich immer noch weiter (derzeit UML 2.0).

Der Entwurf von Klassen und Objekten in der UML

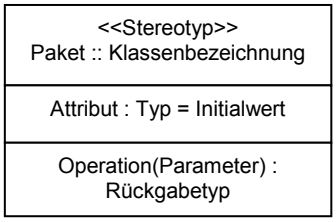
Die Klasse als abstrakter Typ und das Objekt als konkrete Instanz der Klasse stellen in der ooP die Grundlage für ein Softwaresystem dar. Für das Design gibt die UML folgende Regeln vor:

Klassen in der UML

Klasse

Attribut/
Eigenschaft

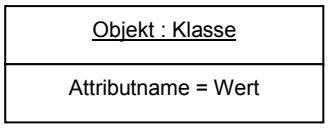
Operation/
Methode



Eine Klasse entspricht einem Plan oder einer Baubeschreibung. Die Definitionen einer Klasse setzt sich aus Attributen und Operationen zusammen.
Dem Klassennamen kann die zugehörige Fachklasse als Stereotyp und der Paketname vorangestellt sein.
Attribute sind Datenelemente, die in jedem Objekt einer Klasse enthalten sind und die für jedes Objekt individuell veränderbar sind.
Objekte kommunizieren über Nachrichten. Sie spiegeln das Verhalten eines Objektes wider. Die Methoden oder Operationen eines Objektes sind eine Sequenz von Anweisungen, die für eine Klasse definiert sind.

Objekte in der UML

Objekt

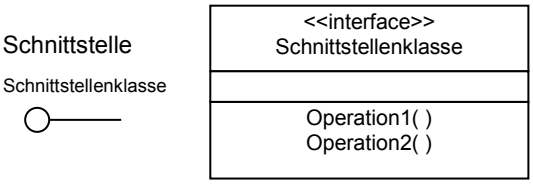


Ein Objekt ist ein Abbild einer Klasse im laufenden System. Die „Klassenkopie“ zur Laufzeit trägt alle definierten Eigenschaften der Klasse und ist durch die Interaktion mit anderen Objekten und dem Zustand der Attribute eindeutig.
Die Bezeichnung eines Objekts wird unterstrichen, um eine Verwechslung mit einer Klasse zu vermeiden

Schnittstellen

Schnittstelle

Schnittstellenklasse



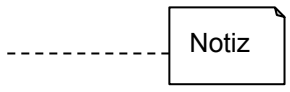
Schnittstellen sind besondere Klassen mit dem Stereotyp „interface“. Sie dienen als „Baugerüst“ für eine Klasse.
Interface - Klassen besitzen keine Attribute, sie stellen den Klassen lediglich Methodendeklarationen zur Verfügung.
Klassen, die nach einer Schnittstellenklasse deklariert sind, entsprechen der Umsetzung eines Interfaces.

Einschränkungen und Notizen

constraint { z.B.: Attribut = 4 oder 5 ...}

Einschränkungen werden neben das zugehörige Attribut gestellt.

Notiz



Notizen heftet man mit einer gestrichelten Linie an das Attribut bzw. die zugehörige Methode. Notizen stehen außerhalb der Klasse. Der Inhalt einer Notiz ist formatfrei (z.B. Text oder Grafik). Sowohl Einschränkungen als auch Notizen helfen, ein Diagramm verständlich zu gestalten.

Die Symbolic der Unified Modelling Language (1)

Klasse
(Bauplan)

Objekt
(Konkrete Instanz)

Moebel : Tisch
seinModell : const char* seinHolz : const char* seinHersteller : const char* seinPreis : double = 0.0
GetPreis() : double SetPreis(pr : double) : void

derTisch : Tisch
seinModell = "Esszimmertisch" seinHolz = "Akazie" seinHersteller = "Holzwurm AG" seinPreis = 399.90

Person
ihrAlter : int {ihrAlter >= 0}
getAlter() : int setAlter(alter : int) : void

Hans : Person
ihrAlter = 32