

Das Objektorientierte Paradigma

Inhalt

Das Objektorientierte Paradigma	1
Inhalt	1
Grundbegriffe	1
Objekte	2
Klassen	2
Prinzipien	2
Abstraktion	2
Geheimnisprinzip	2
Private Methoden	2
Vererbung	3
Einfachvererbung und Spezialisierung	3
Mehrfachvererbung	3
Polymorphismus	3
Überschreiben	3

Web Content Management auf Basis von
XML

b.i.b Dresden im September/Okttober 2000

© alle Rechte bei H.-J. Günther

Grundbegriffe

Die klassische, imperative Programmiertechnik (C, FORTRAN, Pascal...) betrachtet Daten und Funktionen als voneinander getrennt. Auch relationale Datenbanksysteme (RDBMS) teilen diesen Ansatz. Die objektorientierte Sichtweise überwindet diese Einstellung und führt somit zu einer konsequenteren Betrachtungsweise, welche sich durch den gesamten Entwurf bis zur Anwendung der Software durchsetzt. Objektorientierte Programmierung ist demnach nur eine klassische Programmierung, bei der man sich der stilistischen Übung zur maximalen Normierung unterwirft, indem man Daten und ihre Verarbeitung miteinander verschmelzen lässt.

Objekte

Objekte sind aktive Datenstrukturen. Diese Definition beruht auf der Überlegung, dass ein Objekt als Zusammenfassung von Daten und den zu Ihrer Verarbeitung notwendigen Methoden betrachtet werden kann.

Klassen

Klassen sind die Baupläne für Objekte. Diese Definition leitet sich aus Überlegungen zu abstrakten Datentypen her. Somit gehört in Umkehrung jedes Objekt einer Klasse an. Klassen definieren die grundlegenden Verhaltensweisen von Objekten sowie deren Attribute. Man nennt ein Objekt eine Instanz einer Klasse und eine Klasse entsprechend eine Abstraktion eines Objektes.

Eine abstrakte Klasse beinhaltet Definitionen von Eigenschaften darunter liegender Klassen - der Unterklassen (sub classes). Oberklassen (Superklassen) bilden Zusammenfassungen der gemeinsamen Methoden und/oder Attribute von Unterklassen. Prinzipiell kann eine Klasse gleichzeitig Ober- und Unterklasse sein.

Prinzipien

Die Prinzipien der objektorientierten Programmierung beschreiben die notwendigen Begriffe zur Handhabung von Objekten und Klassen. Sie unterscheiden sich von den klassischen Programmierprinzipien nur darin, dass sie die alt hergebrachten Prinzipien in höchst normierter Weise umsetzen und so eine leistungsfähige Methodik für abstraktes Arbeiten auf rein softwaretechnologischer Basis verfügbar machen.

Abstraktion

Abstraktion ist ein theoretisches Konzept zur Beschreibung von Objekten und Klassen. Da ein Objekt sowohl Daten (Attribute), als auch Methoden beinhaltet, kann die Abstraktion sowohl die Daten selbst als auch deren Attribute konsistent behandeln. Die abstrakte Beschreibung von Objekten und Klassen dient im Wesentlichen zur begrifflichen Definition derselben.

Geheimnisprinzip

Das Geheimnisprinzip (Information Hiding) verlangt, dass Methoden in einem Objekt so gestaltet werden, dass ausschließlich ein Methodenname und eventuelle Übergabeparameter bekannt sein dürfen um die Methode zu nutzen. Die exakte Realisierung der Methode sowie für die Abarbeitung nötige und im Objekt abgelegte Daten dürfen nicht nach außen nicht in Erscheinung treten. Damit wird das Objekt bzw. die Klasse zur „Black Box mit Schaltern und Lämpchen“.

Private Methoden

Methoden, die ausschließlich innerhalb eines Objektes von Bedeutung sind und von keinem anderen Objekt aufgerufen werden, heißen „private Methoden“. Diese haben keine externe Schnittstelle, übernehmen oder übergeben keine Daten über das sie kapselnde Objekt hinaus und ihre Namen sind nach außen ebenso unbekannt wie ihre konkrete Implementation.

Vererbung

Vererbung ist die Modifikation einer Klasse ohne Manipulation ihrer Implementation. In dieser Definition wird eingeräumt, dass ein Objekt als Klasse betrachtet wird, wenn es selbst als Parent für eine Vererbung genutzt wird. Diese begriffliche Übertragung wird unter dem Aspekt als zulässig betrachtet, dass ein Parent Objekt prinzipiell das Verhalten seiner Child Objekte, wenn auch unvollständig, beschreibt. Man mache sich diesen Unterschied bewusst!

Einfachvererbung und Spezialisierung

Simple Inheritance (einfache Vererbung) beschreibt den Vorgang der Manipulation einer einzelnen Klasse durch Hinzufügen von neuen oder Sperren von bestehenden Methoden ohne Eingriff in ihre Implementation. Durch einfache Vererbung entsteht aus einer Klasse (Super class) eine Unterklasse (Sub class). Diese Definition betrachtet auch Objekte als Klassen im Sinne einer Beschreibung ihrer Child Objekte. Der Vorgang der Unterklassenbildung heißt Spezialisierung.

Mehrfachvererbung

Die Mehrfachvererbung (Multiple Inheritance, multiple Vererbung) beschreibt den Vorgang der Manipulation mehrerer Klassen durch Kombination und/oder Ergänzen und/oder Sperren von Methoden, deren Ursprung in diesen verschiedenen Klassen zu suchen ist. In diesem Falle hat also eine Sub class mehrere Parents, die als Superklassen genutzt werden. Das Ergebnis einer Mehrfachvererbung ist in der Regel, aber nicht zwangsläufig, eine einzige neue Klasse. Die Verwendung des Begriffs der Klasse wird hier der Konsistenz wegen gewählt, wenngleich Mehrfachvererbung in praxi meistens auf gewöhnliche Objekte angewendet wird.

Ein häufiges Problem bei der Mehrfachvererbung ist der Namenskonflikt bei Verwendung gleich benannter Methoden oder Daten mit Ursprung in verschiedenen Objekten und/oder Klassen. Dieser Konflikt wird spätestens bei der Bildung der jeweiligen Sub classes offenbar und muss vom Entwickler (halb)manuell aufgelöst werden.

Ein weiteres Problem der Mehrfachvererbung ist die Verwendung unterschiedlicher Bezeichner für das selbe Attribut oder, seltener, die gleiche Methode. Dieser Konflikt wird oft erst bei der Nutzung der neu entstandenen Unterklasse offenbar, da er keinen Widerspruch zum objektorientierten Paradigma darstellt und mithin als ursächlich inhaltliches Problem nicht automatisch erkannt werden kann.

Die Lösung der Probleme der Mehrfachvererbung liegt einzig in korrektem Entwurf und sauber dokumentierter Implementation von Klassen. Ein weiteres Hilfsmittel ist die pseudo multiple Vererbung. Hier wird der prinzipiell in einem Schritt realisierbare Vorgang der Mehrfachvererbung durch Referenzieren der Superklassen in einem Objekt und erst anschließende (indirekte) Vererbung der Referenzen umgesetzt. So treten die meisten Probleme bereits bei der Referenzierung auf und können so automatisch erkannt und behoben werden. Diese Methode wird jedoch häufig nicht konsequent objektorientiert umgesetzt, da sie Listen von Referenzen (z.B. Links) nutzt, welche die direkte Kopplung der Objekte und Klassen unterbricht. Diese Verletzung des Paradigmas wird jedoch durch die Sicherung des Entwicklungsprozesses aufgewogen.

Polymorphismus

Polymorphismus (Vielgestaltigkeit) bezeichnet die Möglichkeit, gleichnamige Methodenbezeichner bei unterschiedlicher und/oder gleicher Funktionalität in verschiedenen Klassen zu verwenden, wobei der Aufruf der gleichnamigen Methoden differieren kann. Dies dient in der Regel dazu, einem Methodenbezeichner zur Laufzeit die konkrete Ausprägung der dahinter steckenden Methode mitzuteilen und so die Mehrdeutigkeit ähnlicher Prozesse beschreiben und steuern zu können. Diese Zuordnung zur Laufzeit heißt dynamisches Binden oder late binding. Man mache sich bewusst, dass Polymorphismus selbst auch polymorph sein kann. Weiterhin sei bemerkt, dass Polymorphie nicht zwangsläufig auch Abstraktion beinhalten muss. Es sei jedoch empfohlen, im Entwurf rein formal die Abstraktion einzubinden. Dies sichert die syntaktische Unabhängigkeit.

Überschreiben

Das Überschreiben (Overriding) ist ein Vorgang mit 2 Teilschritten. Zuerst wird durch Vererbung eine Instanz erzeugt. Diese wird dann durch Überschreiben der Werte von Attributen oder Methoden unter Beibehaltung der durch die Vererbung erhaltenen Bezeichner modifiziert. Das Hinzufügen von Attributen und/oder Methoden zur Instanz ist hierbei zulässig. Man mache sich bewusst, dass Overriding dem Abstraktionsprinzip widerspricht! Es ist folglich sehr sparsam zu verwenden und, insbesondere im Entwurf, möglichst zu umgehen. Eleganter, wenn auch zuweilen erheblich aufwendiger, ist die Neudefinition von Methoden bzw. Attributen in der durch Vererbung erhaltenen Instanz. Die Entscheidung ist vom Softwarearchitekten in Abhängigkeit vom konkreten Problem und dem zu verwendenden System zu treffen, was wiederum zu einer Verletzung der Souveränität des Softwaredesigns führt.