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

Konzepte der objektorientierten Programmierung

Gedanken zur Programmierung!?

„Sieh Dir das Wesen der Dinge an,
indem Du sie in Materie, Gestalt und Zweck teilst."

„Prüfe die Beschaffenheit der Gestalt eines
Objekts und trenne sie von seiner Materie ab."

- Kaiser Marc Aurel, ca. 170 n. Chr. -

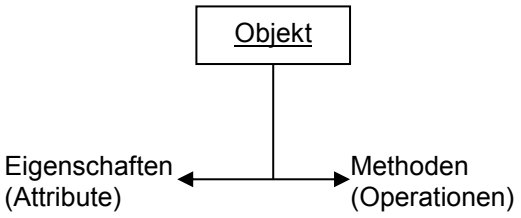
Der Kern der Aussage besteht in der Abstraktion, ein mächtiges, wenn nicht das mächtigste Werkzeug des menschlichen Erfindungsgeistes bei der Softwareentwicklung.

☞ **Erläutern Sie die Bedeutung der Abstraktion bei der Software - Entwicklung!**

Objektorientierte Programmierung

☞ **Welche Vorteile bringt dieses „Baukastenprinzip" bei der Softwareproduktion?**

Der Gedanke der Wiederverwendbarkeit von Software spielte bei der Entwicklung der objektorientierten Programmierung eine große Rolle. Dazu müssen Programme aus einzelnen Komponenten (Objekten) mit bekannten Eigenschaften (Attributen) bestehen, die je nach Bedarf zusammengefügt werden können. Jedes dieser Objekte verfügt über bestimmte Methoden (Operationen), die es handlungsfähig machen.
Dieses Modell entspricht der Hardware – Welt. Beim Bau eines Computers erfindet man die einzelnen Komponenten (z.B. CPU, Motherboard, Festplatte...) nicht jedes Mal erneut, sondern fügt sie entsprechend den gewünschten Anforderungen des Kunden zusammen.



Klasse und Objekt

☞ **Erläutern Sie den Unterschied zwischen Klasse und Objekt!**

In der ooP unterscheidet man zwischen Klasse und Objekt. Die Klasse ist der Bauplan, in dem beschrieben wird, welche Eigenschaften (Attribute) und Operationen (Methoden) Objekte dieser Klasse besitzen. Objekte sind konkrete Exemplare (Instanzen) einer Klasse.

Klasse

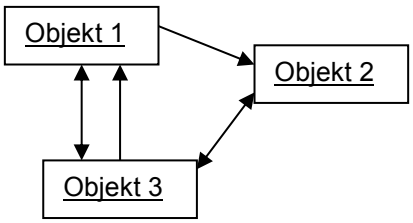
Mitarbeiter
Pers.- Nr. Abteilung Konto ...
Pers.- Nr. anzeigen( ) ...

Objekt

Hans Muster: Mitarbeiter	
Pers.- Nr.	1
Abteilung	1
Konto	DE01 5763 2075 00
...	

Auf die Daten (Attribute) eines Objekts kann nur über definierte Schnittstellenfunktionen (Methoden) zugegriffen werden (vgl. Kapselung und Verbergen von Daten!).

☞ **Diskutieren Sie, wie das Prinzip der Objektorientierung in Programmen umgesetzt werden kann!**

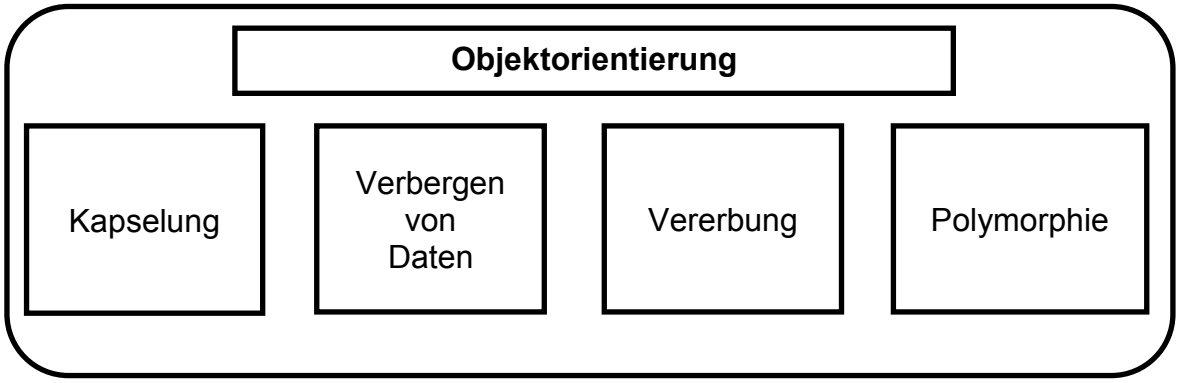


Objekte stehen nicht isoliert, sondern arbeiten über genau definierte Schnittstellen zusammen. Sie versenden Botschaften an weitere Objekte, ändern ihren Status oder beeinflussen die Eigenschaften anderer Objekte.
Erst der Nachrichtenfluss zwischen den Objekten ermöglicht im Zusammenspiel die Simulation komplexer Systeme, ein Software - Programm.
Ziel dabei ist es, die gestellten Anforderungen aus der Realität möglichst wirklichkeitsgetreu nachzustellen.

☞ **Übertragen Sie das Prinzip der objektorientierten Programmierung auf die Komponenten eines Autos!**

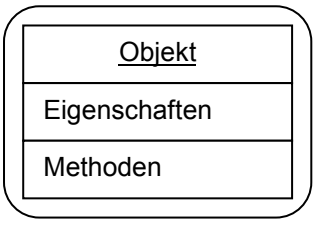
Die vier Säulen der Objektorientierung

C++ unterstützt vollständig die objektorientierte Programmierung (ooP), einschließlich der vier Säulen der objektorientierten Entwicklung:



Kapselung

Wenn ein Techniker einen neuen Rechner zusammenbaut, verschaltet er einzelne Bauteile – etwa CPU, Motherboard, Festplatte, Grafikkarte usw. Eine Festplatte hat bestimmte Eigenschaften und kann bestimmte Verhaltensweisen realisieren. Sie stellt ein komplettes Bauteil mit klar abgegrenzten Aufgaben dar. Alle Eigenschaften der Festplatte (z.B. Zugriffszeit, Speicherkapazität) und möglichen Methoden (lesen, schreiben) sind in diesem Festplattenobjekt verkapselt und nicht über das ganze Motherboard verteilt.
Kapselung bedeutet also, dass alle Eigenschaften (Attribute) und Funktionen (Methoden), die von dem Objekt ausgeführt werden, in der Definition des Objekts enthalten sind.



Kapselung

Diese Vorgehensweise bietet in der Software - Technik entscheidende Vorteile:

- Komponenten (Objekte) können beliebig ausgetauscht, gewartet oder verbessert werden, solange sich die Schnittstellen nicht ändern.
- Programmfehlfunktionen sind wesentlich leichter einzugrenzen, da eine bestimmte Funktionalität in einem Objekt gekapselt und nicht im ganzen Programm verstreut implementiert wird.

☞ **Erklären Sie die Technik der Kapselung mit Hilfe des Beispielobjekts Grafikkarte!**

Verbergen von Daten

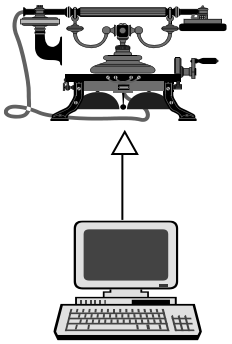
Die CPU eines Computers ist ein äußerst komplex aufgebauter Chip. Kennen Sie die Funktionsweise jedes einzelnen ihrer Schaltkreise? Sicher nicht! Trotzdem arbeiten Sie nahezu täglich mit einem Computer. Das Objekt CPU stellt also eine „black box“ dar. Man kann sie einsetzen, ohne mit ihren internen Abläufen vertraut zu sein.

black box

In der objektorientierten Programmierung ist es nicht notwendig, dass der Benutzer eines Objekts die Funktionsweise aller Daten und Methoden eines Objekts kennt. Er muss sie nur anwenden können. Die Daten sind vor ihm verborgen.
C++ unterstützt die Kapselung und das Verbergen von Daten über benutzerdefinierte Typen, die sogenannten Klassen. Die eigentliche innere Funktionsweise der Klasse ist nicht sichtbar. Der Benutzer muss nicht wissen, wie eine Klasse funktioniert, er muss sie nur verwenden können.

☞ **Beschreiben Sie das Verbergen von Daten mit Hilfe des Beispielobjekts Arbeitsspeicher!**

Vererbung

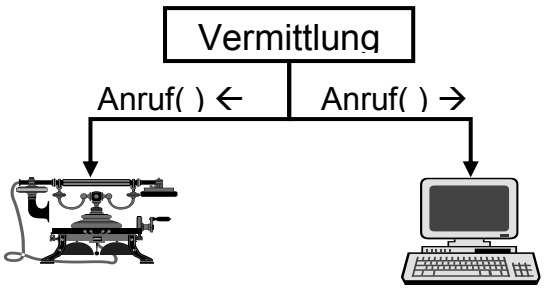


Vererbung bedeutet, dass ein Objekt auf den Eigenschaften eines anderen Objekts aufbauen kann. Es erbt quasi die Eigenschaften und Methoden des Basisobjekts.
Nehmen wir als Beispiel ein Bildschirmtelefon. Bei einem eingehenden Anruf schaltet sich der Bildschirm ein und eine Stimme sagt: „Sie werden angerufen!“. Das Bildschirmtelefon ist kein komplett neues Objekt sondern baut auf bekannte Eigenschaften (eigene Nummer...) und Methoden (wählen, Anruf entgegen nehmen...) eines normalen Telefons auf. Das Bildschirmtelefon besitzt also gegenüber einem Telefon lediglich einige neue Merkmale.
C++ unterstützt den Grundgedanken der Wiederverwendbarkeit durch Vererbung. Man kann einen neuen Typ (Klasse) deklarieren, der eine Erweiterung eines vorhandenen Typs darstellt. Man sagt, dass diese neue Unterklasse von einem vorhandenen Typ abgeleitet ist und spricht auch von einer Basis- bzw. einer abgeleiteten Klasse.
Das Bildschirmtelefon ist vom guten alten Telefon abgeleitet und erbt dessen gesamte Eigenschaften. Bei Bedarf kann man aber neue hinzufügen.

☞ **Welche Vorteile bietet die Vererbung bei der Softwareproduktion?**

Polymorphie

Polymorphie stammt aus dem griechischen und bedeutet „vielgestaltig“. Viele Objekte können dieselbe Methode verwenden und unternehmen genau die richtigen Schritte, um die Funktion auszuführen.



Als Beispiel dient wieder das vorher beschriebene Bildschirmtelefon. Bei einem eingehenden Ruf (Methode) verhält sich das Bildschirmtelefon nicht wie gewohnt. Es klingelt nicht, sondern der Bildschirm schaltet sich ein und eine Stimme sagt: „Sie werden angerufen!“. Die Telefonvermittlung weiß hiervon jedoch nichts – sie sendet keine speziellen Signale für einzelne Telefonarten. Von der Vermittlungszentrale geht einfach ein Ruf (Methode) aus – das normale Telefon klingelt, ein elektronisches Telefon trillert und das Bildschirmtelefon „spricht“. Jedes Telefon unternimmt genau das Richtige, da es die von der Telefonvermittlung kommende Nachricht interpretiert.

☞ **Erklären Sie Polymorphie mit Hilfe eines Ausgabebefehls an einen Bildschirm und einen Drucker!**

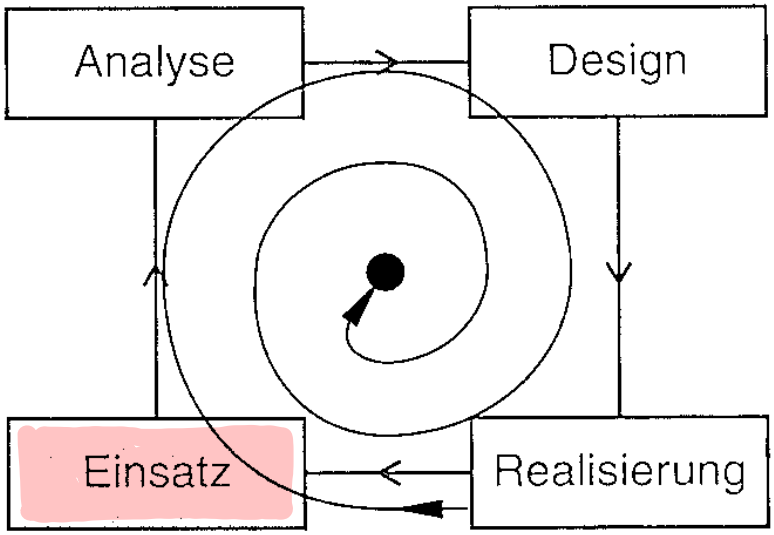
Objektorientierter Programmentwurf nach der Unified Modelling Language (UML)

Mit der Objektorientierung soll die Komplexität der Software - Entwicklung erleichtert werden. Objekte sind geschlossene, kleine Programmeinheiten, die in Beziehung zu anderen Objekten stehen. Im objektorientierten Programmentwurf kommt es darauf an, die Objekte möglichst allgemeingültig (für Programme wiederverwendbar) definieren zu können und die Beziehungen der Objekte untereinander zu bestimmen.

1. Grundlegende Projektzyklen des "Spiralmodells":

Ein objektorientiert durchgeführtes Softwareprojekt wird in unterschiedlichen **Entwicklungsschritten** (Projektzyklen) durchlaufen. Für jeden dieser Schritte stellt die UML Vorgehensweisen und Entwurfsmodelle zur Verfügung. Jeder nachfolgend dargestellte Projektzyklus kann mehrfach durchlaufen werden.
Das Projektmanagement sollte nach den Methoden der Teamarbeit realisiert werden.

☞ **Beschreiben Sie den Ablauf eines objektorientiert durchgeführten Softwareprojekts!**



Analyse: Die Softwarelösung wird mit den künftigen Anwenderinnen und Anwendern in einem Pflichtenheft abgestimmt. Hier werden die problem- und softwarebezogene Ebene zusammengeführt.

Design: Das in der Analyse beschriebene Softwareproblem wird in ein objektorientiertes Modell überführt.

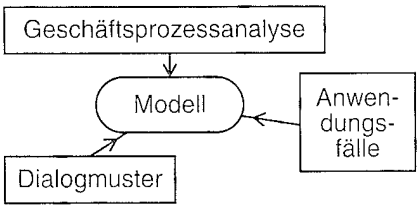
Realisierung: Die Realisierungsphase schließt die Codierung und den Softwaretest ein.

Einsatz: Softwareinstallation und Schulung sind Teil der Einsatzphase. Hier ist auch die Schulung der Mitarbeiterinnen und ein Abgleich mit dem Pflichtenheft angesiedelt.

2. Die Projektzyklen im Detail:

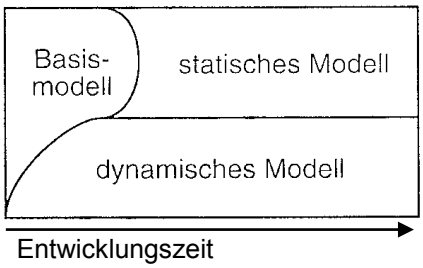
☞ **Beschreiben Sie die Inhalte der einzelnen Projektzyklen!**

2.1 Analyse



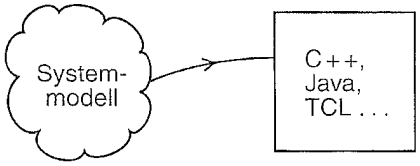
Geschäftsprozessanalyse: Der innerbetriebliche Ablauf, der durch Software unterstützt werden soll, wird analysiert.
Anwendungsfälle: Unterschiedliche typische Szenarien des Geschäftsprozesses werden untersucht und beschrieben.
Dialogmuster: Typische Begriffe, Ausdrucke, Formulare oder betriebliche Vorgehensweisen werden erfasst, so dass die Software später ein Abbild der betrieblichen Wirklichkeit wird.

2.2 Design



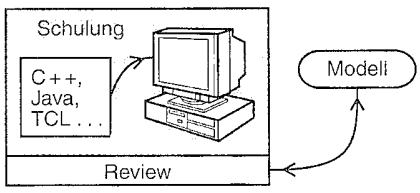
Basismodell: Im Basismodell werden die Grundelemente Klassen, Objekte, Attribute und Operationen entworfen.
Statisches Modell: Dieses Modell beschreibt die Beziehungen der Klassen und Objekte, die Ergebnis des Basismodells waren.
Dynamisches Modell: Das dynamische Modell beschreibt den Zustand und die Reaktion der Objekte auf Aktivitäten zur Laufzeit. Die Beschreibung kann in der UML in unterschiedlichen Diagrammen erfolgen. Diese sind das Sequenz-, Kollaborations-, Zustands- oder Aktivitätsdiagramm.

2.3 Realisierung



Kodierung: Die Kodierung sollte in einer objektorientierten Programmiersprache erfolgen, da der Entwurfsansatz so direkt programmtechnisch umsetzbar ist. Beispiele sind C++, Java, Smalltalk, C#, VB.NET oder die Scriptsprache TCL.
Softwaretest: Beim Softwaretest wird kontrolliert, ob die Ergebnisse der Designphase entsprechend umgesetzt wurden.

2.4 Einsatz



Installation: Das Produkt wird auf den Rechnern des Auftraggebers implementiert.
Schulung: Die Mitarbeiter erhalten eine Einweisung, um mit dem Produkt arbeiten zu können.
Review: Ein Vergleich des tatsächlichen Endproduktes mit dem geplanten Produkt zeigt, ob Änderungen notwendig sind.

3. Fallbeispiel Lohnbuchhaltung

☞ **Erklären Sie die einzelnen Projektzyklen am Beispiel der Lohnbuchhaltung eines mittleren Betriebes!**