Einführung in die Informatik

Klaus Knopper

Definition von Objektorientierung

Unter Objektorientierung versteht man ein Paradigma für

- die Analyse
- den Entwurf
- und die Implementierung

von objektorientierten Systemen.

Teilaspekte sind die die objektorientierte Analyse (OOA), Design (OOD) und Programmierung (OOP).

Was ist ein Objekt? (1)

Objekte sind physikalische oder konzeptuelle Dinge. Ein System besteht aus vielen Objekten.

Ein Objekt hat ein definiertes Verhalten.

Das Verhalten setzt sich zusammen aus einer Menge genau definierter Operationen zur Erfüllung von Einzelaufgaben. Eine solche Operation wird i.d.R. beim Empfang einer „Nachricht“ ausgeführt.

Was ist ein Objekt? (2)

- Ein Objekt hat einen inneren Zustand.
  Der Zustand des Objekts ist seine „Privatsache“. Das Resultat einer Operation des Objekts (bei Empfang einer Nachricht) hängt jedoch vom aktuellen Zustand des Objektes ab.

- Ein Objekt hat eine eindeutige Identität.
  Die Identität eines Objekts ist unabhängig von seinen anderen Eigenschaften. Es können mehrere verschiedene Objekte mit identischem Verhalten und innerem Zustand im gleichen System existieren.

Beispiel: Termin-Objekt

Objekt „12. Projektbesprechung“ zur Repräsentation eines Termins

Verhalten des Termin-Objekts:
Reaktion auf Nachrichten wie „gibDatum“, „versendeEinladungen“

Zustand des Termin-Objekts (Eigenschaften):
Datum, Ort, Teilnehmer, geplant/bestätigt

Identität des Termin-Objekts:
Die „12. Projektbesprechung“ besteht unabhängig von ihren Eigenschaften wie: “Datum verschieben“, “Ort verlegen“, “Teilnehmer einladen“.

Objekte und Klassen

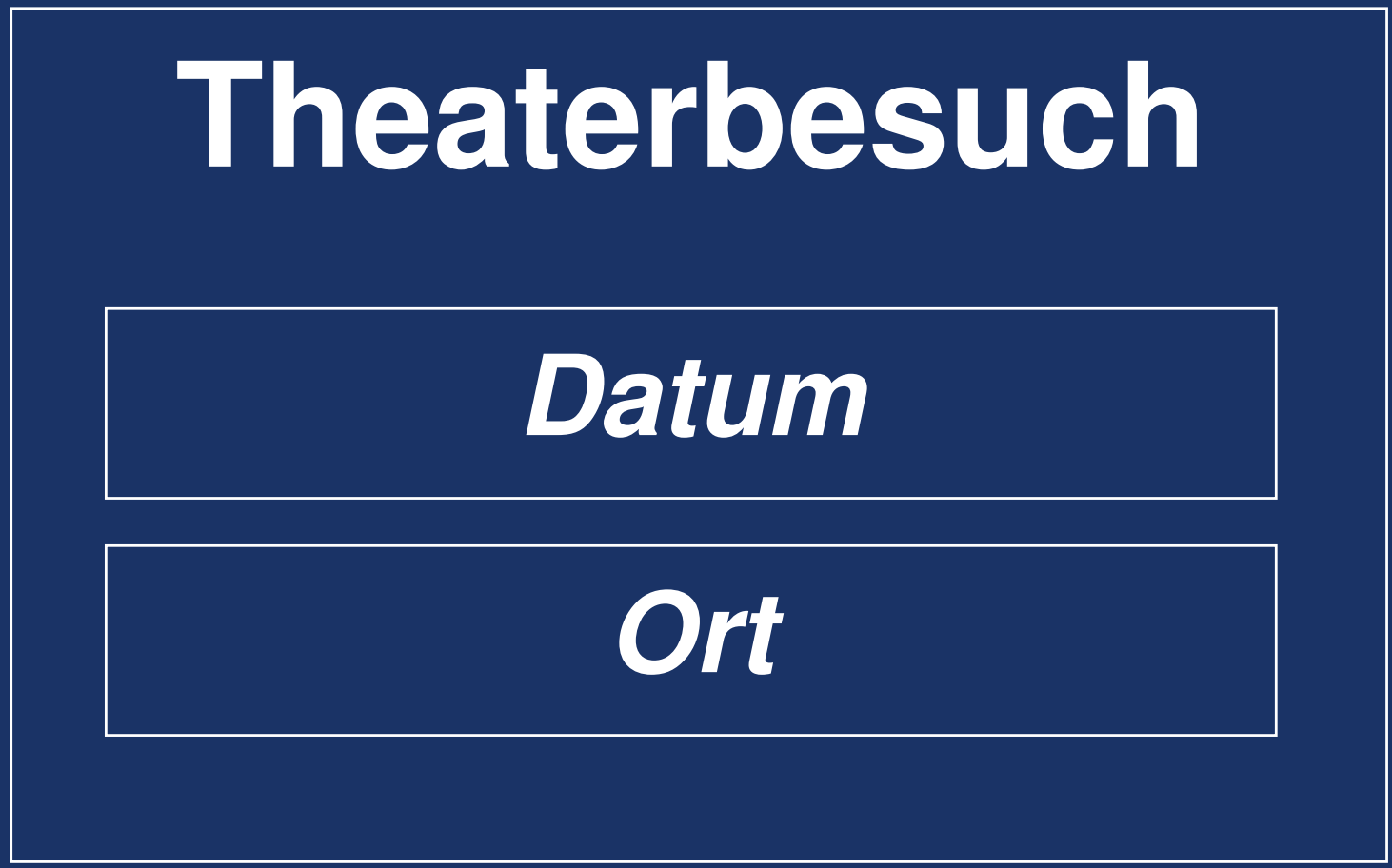
- Jedes *Objekt* gehört zu einer *Klasse*.

  Alle Objekte einer Klasse folgen dem gleichen Verhaltensschema und haben die gleiche innere Struktur. Man sagt: Eine Objekt-Variable ist eine *Instanz* einer Klasse.

- Klassen besitzen einen „Stammbaum“
  in dem Verhaltensschema und innere Struktur durch Vererbung weitergegeben werden. *Vererbung* bedeutet Spezialisierung einer Klasse zu einer Teilklasse.

Eine Nachricht kann verschiedenes Verhalten (verschiedene Operationen) auslösen, je nachdem zu welcher Teilklasse einer Oberklasse das empfangende Objekt gehört (Polymorphie).

Beispiele: Termin-Klassen



Klasse „Theaterbesuch“ mit *Ort* und *Datum* als Eigenschaften.

Objektverwaltung

Eine Klasse weiß ohne Objektverwaltung nicht, welche Objekte sie besitzt bzw. welche Objekte von ihr erzeugt (instanziert) wurden.
Objektverwaltung bedeutet, die Klasse „führt Buch“ über das Erzeugen und Löschen ihrer Objekte. Die Klasse erhält die Möglichkeit, Anfragen und Manipulationen auf der Menge der Objekte der Klasse durchzuführen (class extension, object warehouse). Die Objektverwaltung muss im Entwurf und in der Implementierung realisiert werden.

Geschichte der Objektorientierung (1)

Simula: Ole-Johan Dahl + Krysten Nygaard, Norwegen, 1967
Sprache zur Systemsimulation (d.h. Modellierung!)

Smalltalk: Allan Kay, Adele Goldeberg, H. H. Ingalls, Xerox Palo Alto Research Center (PARC), 1976-1980
Graphische Benutzungsoberflächen für Personal Computing

Seit 1986: ParcPlace Smalltalk und andere Hersteller

Geschichte der Objektorientierung (2)

C++: Stroustrup, Bell Labs (New Jersey), 1984
Erweiterung von C um objektorientierte Konzepte

Eiffel: Bertrand Meyer, 1988
Neudesign einer „sauberen“ Sprache für komplexe Softwaresysteme

Objektorientierte Analyse- und Entwurfsmethoden, ca. 1989-1992
Booch, Coad/Yourdon, Rumbaugh et al., Shlaer/Mellor, Coleman et al.

Java: Ken Arnold, James Gosling, Sun Microsystems,1995

Unified Modeling Language (UML): Rational Corp., 1996-97

Prinzipien der Objektorientierung

Dazu gehören:

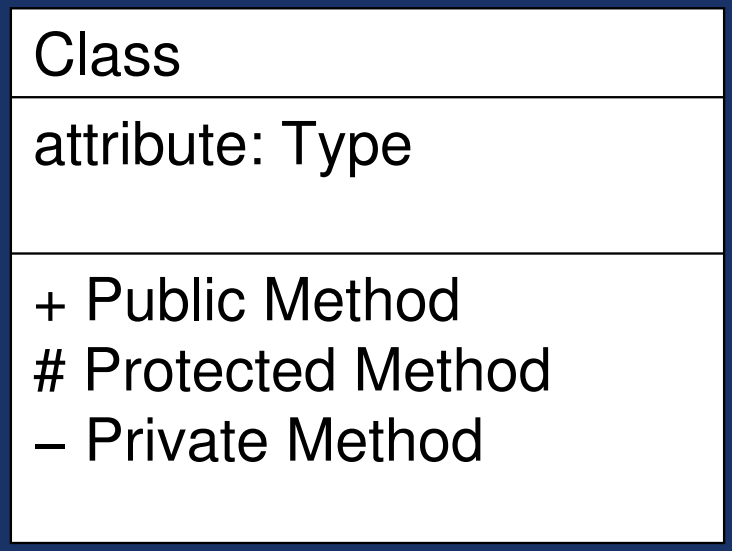
- Abstraktion
- Kapselung
- Wiederverwendung
- Beziehungen
- Polymorphismus

Abstraktion

Darunter versteht man die Trennung von Konzept und Umsetzung, indem man zwischen Objekten und Klassen unterscheidet. Von Klassen gebildete Objekte müssen instanziert werden, um ihre Attribute und das Verhalten zu nutzen, Klassen fassen Objekte des gleichen Typs zusammen.
Mit Hilfe von Klassen und Objekten ist es möglich, die Details einer Problemstellung zu ignorieren und sich auf die grundsätzlichen Gegebenheiten zu konzentrieren. Dadurch ist eine Reduzierung der Komplexität möglich.
Es findet also eine Abbildung der komplexen realen Welt in eine abstrakte logische Ebene (dargestellt durch Objekte und Klassen) statt.

Kapselung

Eine Klasse umfasst eine Menge von Daten (Attributen) und die darauf operierenden Funktionen (sog. *Methoden*). Diese Methoden enthalten Programmcode und dienen als Schnittstelle zur Kommunikation mit anderen Objekten. Die Zusammenfassung von Attributen und Methoden eines Objekts bezeichnet man als Kapselung.



Wiederverwendung

Die Prinzipien der Abstraktion und Kapselung können für die Wiederverwendung von Programmcode genutzt werden. Damit soll die Effizienz gesteigert und die Fehlerrate reduziert werden.

Ein Beispiel sind die sog. Collections. Dies sind Objekte, die andere Objekte anderen Typs verarbeiten können. Damit kann man z.B. sehr effizient Aufgaben wie das Sortieren und Suchen von Elementen durchführen.

Beziehungen

Objekte werden meist nicht isoliert betrachtet, sie stehen meist in Verbindung mit anderen Objekten.
Es gibt versch. Arten von Beziehungen (sog. Assoziationen) zwischen Objektklassen. Eine Assoziationsart ist z.B die „Teile/Ganzes-Beziehung“, dabei setzt sich ein Objekt aus einer Anzahl anderer Objekte zusammen.
Eine andere Art der Beziehung ist die sog. Vererbungshierachie. Dazu zählen die Generalisierung und die Spezialisierung.

Polymorphismus

Unter Polymorphimus (Vielgestaltigkeit) versteht man die Fähigkeit von Objekten, Objekte anderer Klassen und daraus abgeleiteter Klassen aufzunehmen.
Damit kann die gleiche Methode auf verschiedene Objekte der Hierarchie angewendet werden.

Werkzeuge für die objektorientierte Entwicklung (1)

Es gibt diverse Modellierungssoftware für die objektorientierte Analyse (OOA) und das objektorientierte Design (OOD). Im Zuge der ingenieurmäßigen (d.h. systematischen) und rechnergestützten Entwicklung sind viele dieser sog. CASE-Tools (Computer Aided Software Engineering) entstanden. Solch ein CASE-System sollte idealerweise den gesamten Softwarelebenszyklus unterstützten. Alle diese CASE-Werkzeuge basieren heute auf der grafischen Notationssprache UML (Unified Modeling Language), welche mittlerweile eine große Verbreitung hat.

Werkzeuge für die objektorientierte Entwicklung (2)

Alle marktrelevanten Werkzeuge können meist aus den UML-Modellen direkt Programmfunktionen in unterschiedlichen Sprachen generieren. Einige sind zudem in der Lage, den Programmcode zu analysieren und durch Reverse Engineering entsprechende UML-Modelle anzupassen. Werden diese Verfahren systematisch genutzt, spricht man von Round-Trip-Engineering.

Beispiele für CASE-Tools sind Innovator (MID), Rational Rose (IBM) und ArgoUML (Open Source).

Objektorientierte Analyse

Problembeschreibung des Systems
- Pflichtenheft

statische Analyse
- Identifikation von Klassen, Objekten
- Identifikation von Assoziationen zwischen Objekten und Klassen
- Identifikation von Attributen der Objekte und Klassen
- Organisation der Objektklassen z.B. mit Hilfe von Vererbungshierarchien

dynamische Analyse
- Szenarios entwickeln
- Beschreiben des Ereignisflusses: Interaktionsdiagramme
- Zustandsdiagramme entwickeln
- Identifikation der Geschäftsprozesse

Erstellen eines Prototypen der Benutzeroberfläche

Unified Modeling Language (UML)

Die UML ist eine Modellierungsprache, also eine Sprache zur Beschreibung von Softwaresystemen. Es gibt eine einheitliche Notation für die Darstellung des gesamten Softwareprozesses. Dazu werden verschiedene Diagrammtypen und natürlichsprachige Beschreibungen genutzt.
Aktuell ist die Version 2.0, welche auch Programmierkonzepte aktueller Programmiersprachen wie C# und Java unterstützt.

Begriffe in der UML (1)

Artefakt: Ergebnisse oder Produkte eines Entwicklungsprozesses, z.B. eine Klasse oder eine Beschreibung eines Elements

Notiz: An jedes UML-Element kann eine Notiz angehängt werden. Diese beschreibt ein UML-Element näher. Grafisch wird dies durch eine gestrichelte Linie von der Notiz zum zu beschreibenden Element dargestellt.

Szenario: ein Ausschnitt aus dem Gesamtsystem, beschreibt bestimmte Aspekte des Systems

Begriffe in der UML (2)

Constraint: Eine Bedingung oder Einschränkung für die Implementierung eines Elements. Die Bedingung steht in geschweiften Klammern und wird mit einer gestrichelten Linie zum betreffenden Element verbunden.

Classifier: Ein Grundelement der UML, z.B. eine Klasse oder Assoziation. Ein Classifier entspricht einem definierten Element in UML.

Stereotyp: dient zur Zusammenfassung von Elementen in Kategorien. Zum Beispiel kann man Klassen in Entitätsklassen (entity), Oberflächenklassen (boundary) und Controllerklassen (control) einteilen.

Diagrammtypen der UML

Strukturdiagramme:
beschreiben die statische Struktur (zeitunabhängig) von Systemen. Es kann die interne Darstellung der Struktur von Klassen bis hin zur Definition von Architekturen komplexer Systeme genutzt werden.
Dazu zählen z.B. Klassendiagramme, Objektdiagramme, Kompositionsstrukturdiagramme, Verteilungsdiagramme

Verhaltensdiagramme:
beschäftigen sich mit dem dynamischen Verhalten von Systemelementen. Eine Unterkategorie sind die Interaktionsdiagramme, welche die Interaktion zwischen Kommunikationspartnern modellieren.
Beispiele sind Aktiviätsdiagramme, Anwendungsfalldiagramme, Zustandsdiagramme, Interaktionsdiagramme (Sequenzdiagramme, Kommunikationsdiagramme, etc.)

Notation von Objekten und Klassen in UML

Der Objektname:

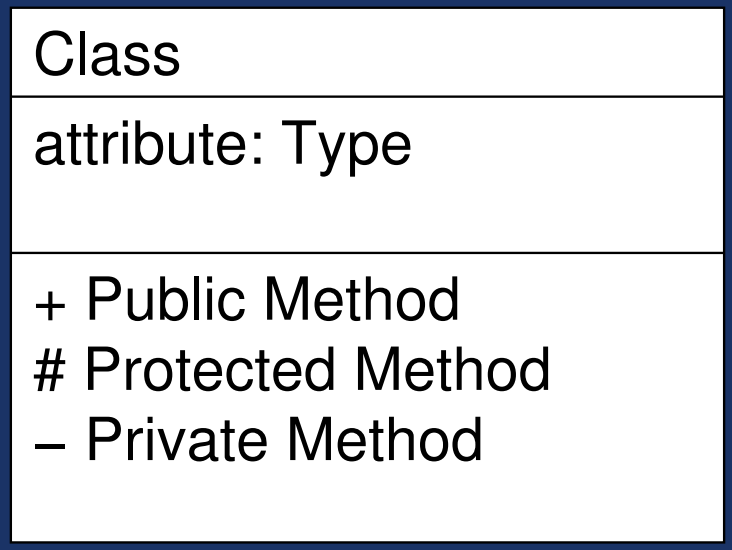
- identifiziert ein Objekt im Objektdiagramm
- muss innerhalb des betrachteten Kontexts (Diagramm) eindeutig sein

Der Klassenname:

- ist ein Substantiv im Singular, ergänzbar durch ein Adjektiv
- ist eindeutig innerhalb des Systems
- optional mit Paketnamen (**`paketname::klassenname`**)

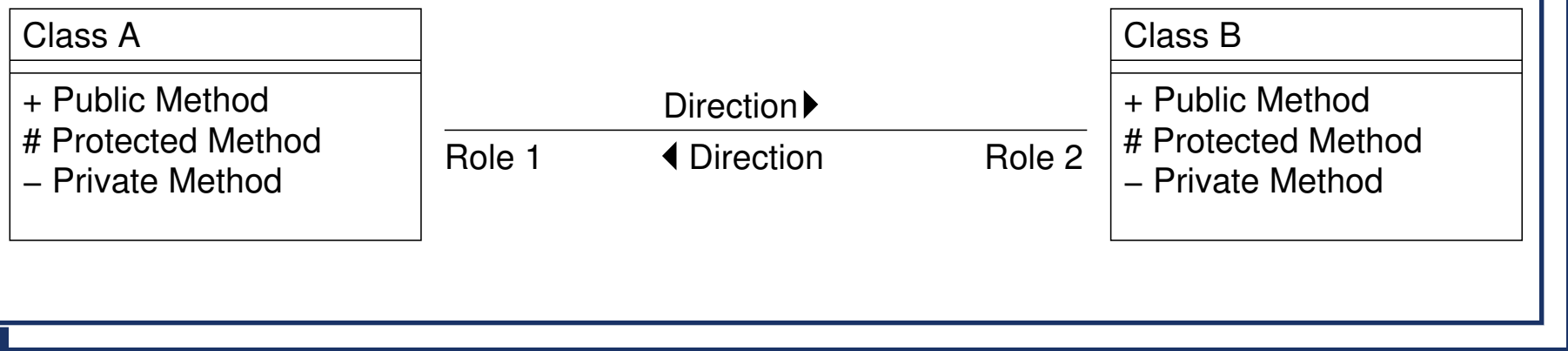
Klassendiagramm

Klassendiagramme stellen die statische Struktur von Systemen dar. Es werden die Klassen mit ihren Eigenschaften (Attributen) und ihrem Verhalten (Operationen) dargestellt. Zudem werden die Beziehungen zwischen den Klassen (Assoziationen) dargestellt.



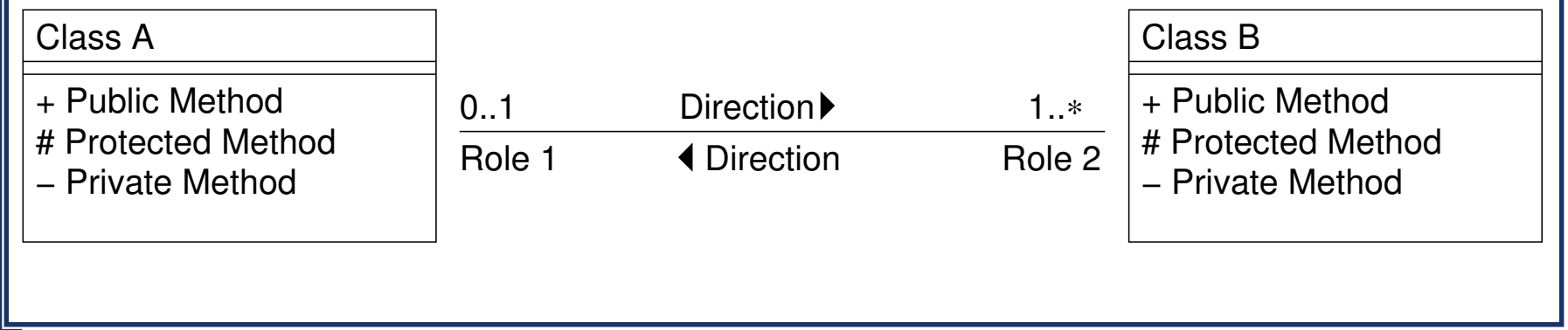
Assoziation

Assoziationen sind die Beziehungen zwischen den Klassen bzw. deren Objekten. Sie werden durch eine Linie zwischen den Klassen dargestellt. Objekte kommunizieren über die Assoziation miteinander. Assoziationen können gerichtet oder ungerichtet sein. Zudem kann die Assoziation einen Namen bekommen, welche die Beziehung zwischen den Klassen beschreibt.



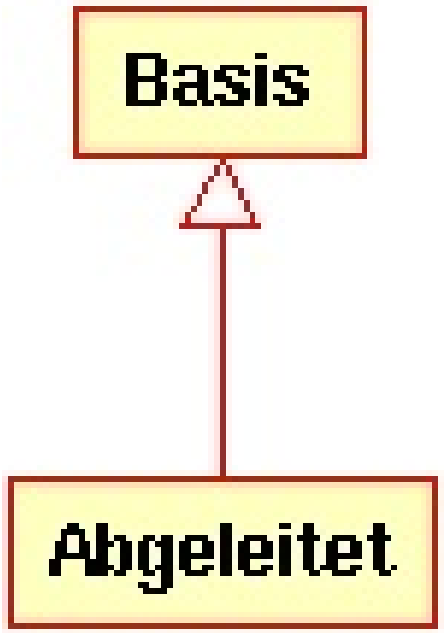
Multiplizitäten

Assoziationen zwischen Klassen können mit sog. Multiplizitäten versehen werden. Diese stellen dar, wieviele Objekte einer Klasse mit wievielen Objekten einer anderen Klassen in Verbindung stehen. Multiplizitäten werden durch eine Zahl auf der Assoziationslinie auf der Seite der betreffenden Klasse dargestellt.



Vererbung

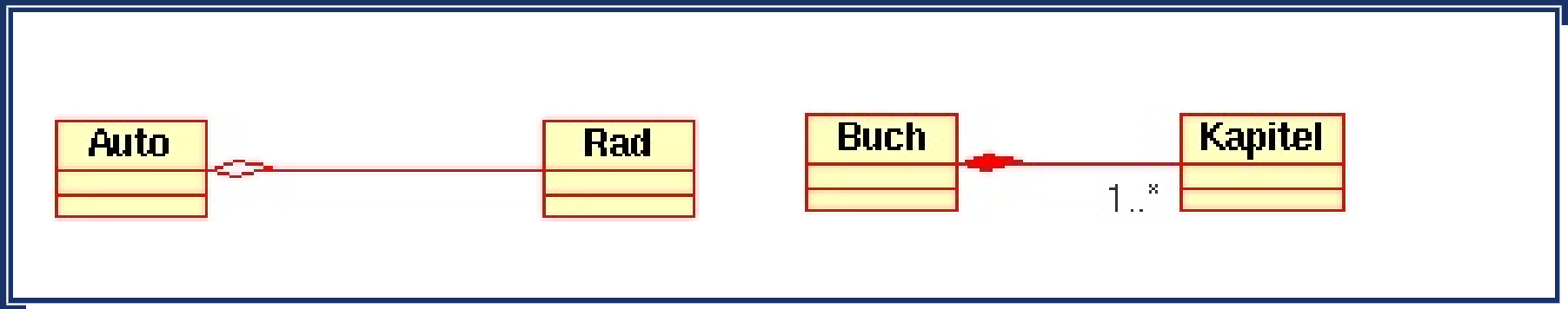
Klassen können über eine Vererbungshierachie miteinander verbunden sein. Dies wird durch einen Pfeil dargestellt, der auf die Oberklasse zeigt. Bei Spezialisierungen bzw. Generalisierungen gibt es keine Multiplizitäten.



Aggregation und Komposition

Mit einer Aggregation kann eine Teile/Ganzes-Beziehung dargestellt werden. Die Aggregation ist eine bestimmte Art der Assoziation. Dargestellt wird diese durch eine Raute an der Seite des Ganzen.

Eine Komposition entspricht einer Aggregation, die Beziehung ist aber existenziell, d.h. die Teile können nur exisieren, wenn das Ganze vorhanden ist. Mit einer ausgefüllten Raute kann eine Komposition dargestellt werden.

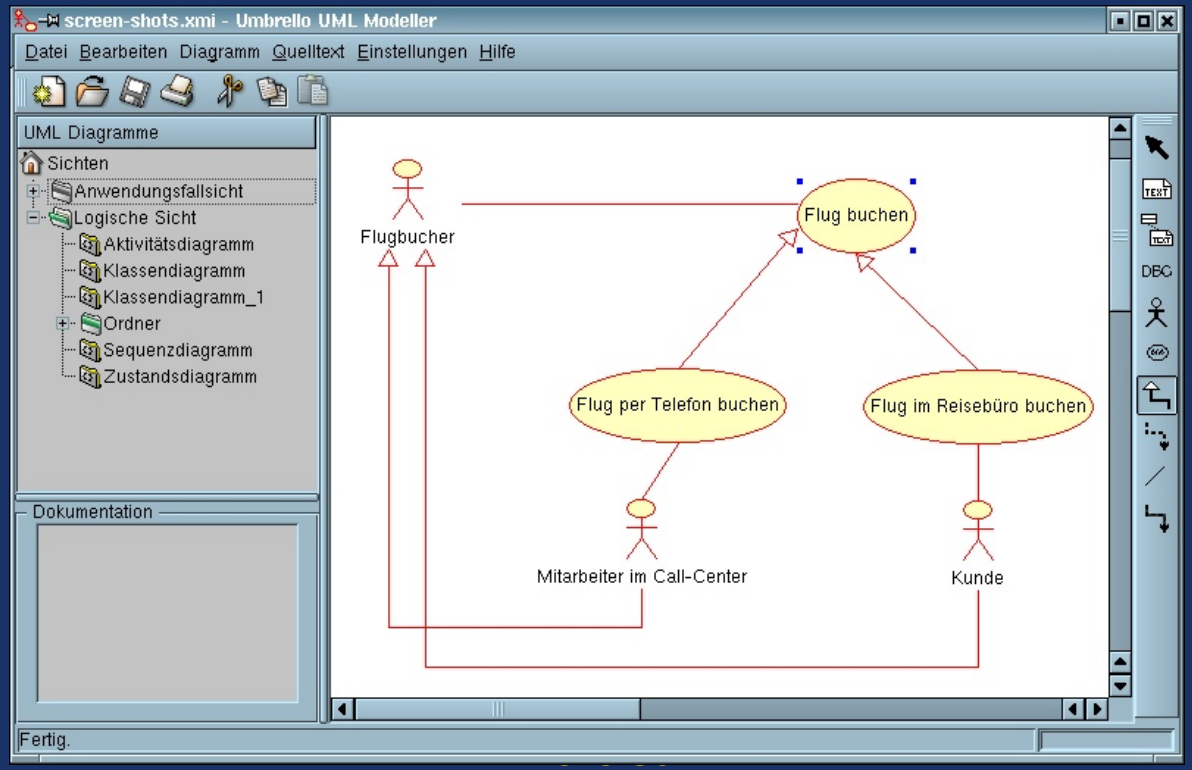


Objektdiagramm

Mit einem Objektdiagramm können die Beziehungen bestimmter Objekte von Klassen dargestellt werden. Es wird somit ein Schnappschuss des Programms zur Laufzeit erzeugt. Zu jedem Klassendiagramm kann mindestens ein Objektdiagramm erstellt werden.

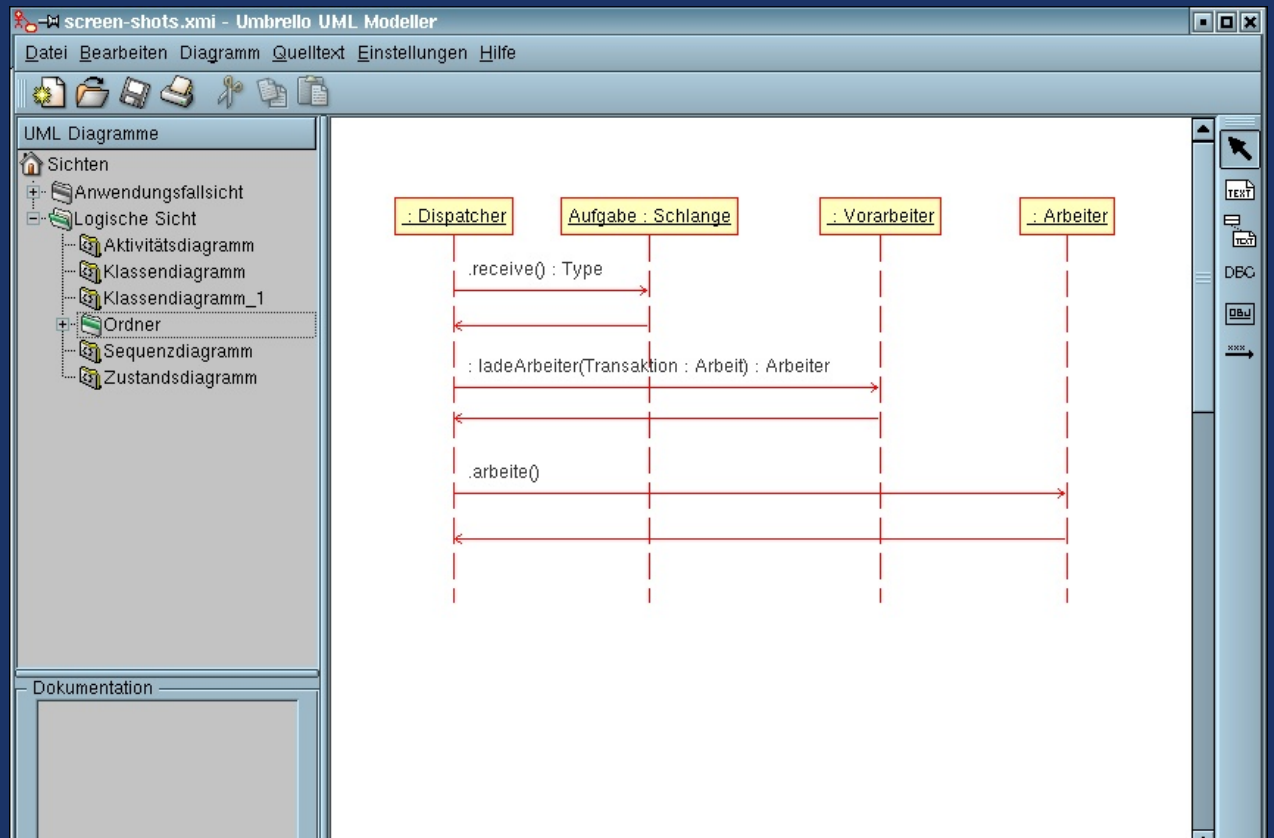
Anwendungsfalldiagramm

In Anwendungsfall- oder sog. use-case-Diagrammen kann das externe Systemverhalten aus Anwendersicht modelliert werden. Ein use case ist ein Geschäftsprozess aus Anwendersicht. use cases werden als Elipsen in einem Kasten, dem System, dargestellt und zu dem dazugehörigen Akteur (dargestellt als Strichmännchen), der dieses Anwendungsfall auslöst, verbunden.



Das Sequenzdiagramm

... stellt die Kommunikation der Objekte in einer bestimmten Szene dar. Die Kommunkation findet über Nachrichten zwischen Objekten statt, welche im Diagramm dargestellt werden. Der Nachrichtenaustausch findet in einer gewissen Reihenfolge statt. Dabei schreitet die Zeit anhand einer Zeitlinie von oben nach unten fort.



opjektorientiertes Design (OOD)

Abbild des späteren Programms
- Beschreibung der Klassen, Methoden etc. in Pseudocode

Statisches Modell
- Erstellen der Architekturklassen
- physikalische Verteilung der Funktionalitäten auf Rechnerknoten

Dynamisches Modell
- übersichtliche Beschreibung der komplexen Objektkommunikation
- Erzeugen von Entwurfsmustern

objektorientierte Programmierung

Im Anschluss an das objektorientierte Design sind alle Einzelheiten des Aufbaus und der Funktionalität sowie Ablauf des Programm bekannt. Mit Hilfe einer Programmiersprache kann nun die Implementierung der Software vorgenommen werden.
Es gibt verschiedene objektorientierte Programmiersprachen (Smalltalk, C++, Java, C#, ...). Die verbreitetste und bekannteste ist Java der Firma Sun Mircosystems.

Beispiel für eine Java-Klasse:

```
static class Element {
      Object inhalt;
      Element nachfolger  = null;
      Element vorgänger = null;
      // Konstruktor-Methode
      Element (Object obj) { inhalt = obj; }
   };
```