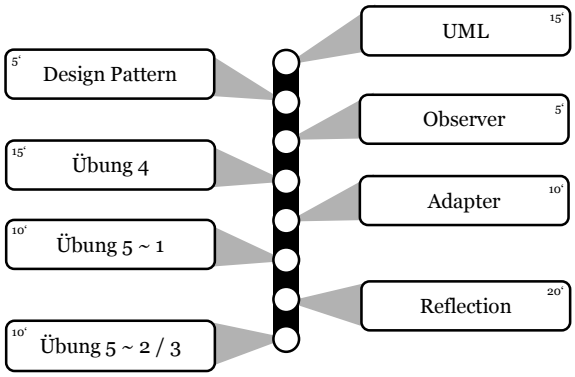
Was wir heute tun

2. Mai 2003



UML

ist eine Sprache & Notation zur Spezifikation, Visualisierung & Dokumentation von Modellen für Softwaresysteme

von den „Amigos" Grady Booch, Ivar Jacobson & Jim Rumbaugh

heutige Softwaresysteme sind komplex,
gar eines der komplexesten Gebilde,
die der Mensch je geschaffen hat

Komplexität ↔ Schwierigkeit

durch intelligente Strukturierung
→ Vereinfachung durch Abstraktion

Klassen in UML

eine Klasse ist die Definition der Attribute, Operationen & der Semantik für eine Menge von Objekten. Alle Objekte einer Klasse entsprechen dieser Definition.

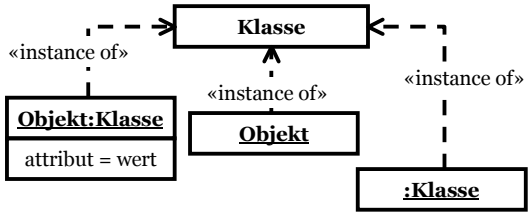
Klasse
attribut
operation()

Paket::Klasse

Abstrakte Klasse

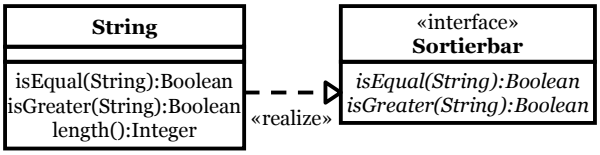
Objekte in UML

ein Objekt ist eine im laufenden System konkret vorhandene & agierende Einheit. Jedes Objekt ist ein Exemplar einer Klasse. Das durch Nachrichten definierte Verhalten gilt für alle Objekte einer Klassen gleichermassen, ebenso die Struktur der Attribute. Die Werte der Attribute sind für jedes Objekt individuell.



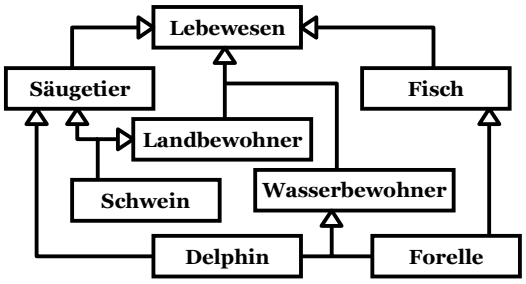
Interfaces in UML

Schnittstellen beschreiben einen ausgewählten Teil des extern sichtbaren Verhaltens von Modellelementen. Schnittstellen definieren ausschliesslich abstrakte Operationen.



Vererbung in UML

Vererbung ist ein Programmiersprachenkonzept für die Relation zwischen Ober- & Unterklasse, wodurch Attribute & Operationen der Oberklasse auch den Unterklassen zugänglich gemacht werden.



Sichtbarkeit in UML

+public element
#protected element
-private element
~package element

Attribute & Operationen in UML

Attribute – Felder, Membervariablen

```
attribut ::== [Sichtbarkeit] Attributname :
              Paket::Typ = Initialwert.
```

Operationen – Methoden, Eigenschaften

```
operation ::== [Sichtbarkeit] Operationsname
               (Parameterliste): RückgabeTyp.
Parameterliste ::== Richtung Name : Typ = Defaultwert.
Richtung ::== in | out | inout.
```

Das war UML *light!*

UML ist noch viel umfangreicher & mächtiger!

wird wirklich in der Praxis eingesetzt bei:
- objekt-orientierter Programmierung
- Datenbankentwurf

UML-Designer Visio in Visual Studio.net

viele gute & schlechte Bücher zum Thema

Offizielle Seite zu UML
`www.omg.org/uml`

Kurzübersicht zu UML 1.4
`http://www.oose.de/downloads/uml-notationsuebersicht.pdf`

Becher
`www.oose.de/uml/becher`

Was ist ein Entwurfsmuster / ein Design Pattern?

> *Jedes Muster beschreibt ein in unserer Umwelt beständig wiederkehrendes Problem & erläutert den Kern der Lösung für dieses Problem, so dass Sie diese Lösung beliebig oft anwenden können, ohne sie jemals ein zweites Mal gleich auszuführen.*

Christopher Alexander, 1977 über Architektur

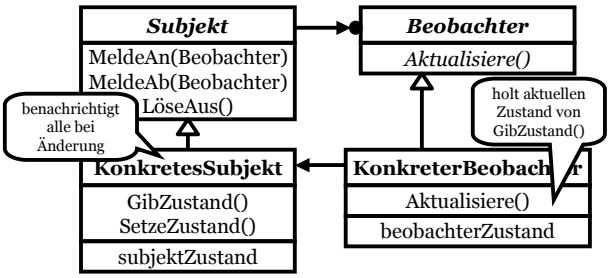
ein Entwurfsmuster besteht aus
vier grundlegenden Elementen:
- Mustername – *Kommunikation*
- Problemabschnitt – *Einsatzgebiet, Kontext*
- Lösungsabschnitt – *abstrakter Entwurf mit Elementen*
- Konsequenzenabschnitt – *Vor- & Nachteile*

Entwurfsmuster helfen wiederum komplexe,
umfangreiche Systeme in den Griff zu bekommen

Observer

www.dofactory.com/Patterns/PatternObserver.aspx

definiere eine 1-zu-n-Abhängigkeit zwischen Objekten, so dass die Änderungen des Zustands eines Objektes dazu führt, dass alle abhängigen Objekte benachrichtigt & automatisch aktualisiert werden.



Adapter

www.dofactory.com/Patterns/PatternAdapter.aspx

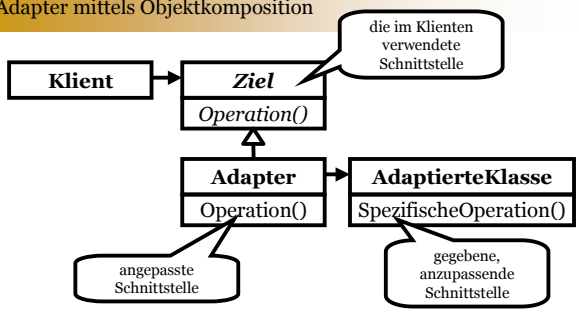
passe eine Schnittstelle einer Klasse an eine andere von ihren Klienten erwartete Schnittstelle an. Das Adapter-muster lässt Klassen zusammenarbeiten, die wegen inkompatibler Schnittstellen ansonsten nicht dazu in der Lage wären.

zwei Varianten:

Implementation mittels

- Mehrfachvererbung
- Objektkomposition

Adapter mittels Objektkomposition



→ Komplexizität des Adapters hängt von der Ähnlichkeit der Zielschnittstelle mit der Schnittstelle der anzupassenden Klasse ab

Builder im Sinne von Quellcodegenerator

```
public string CreateCollectorConnection(
 string sourceClass, string destClass )
{
 // ... inhalt bestimmen
 return "class GenerierteKlasse {\n" + inhalt + "\n}";
}
```

die Methode erzeugt Quellcode für eine neue Klasse

→ um benutzt zu werden, muss die dann aber noch kompiliert werden

die Parameter der Methode enthalten die Namen der Quell- & Zielklasse

wie komme ich an die Struktur dieser implementierten & kompilierten Klassen?

Reflection http://www.galileocomputing.de/openbook/csharp/kap31.htm#Xxx719559

Reflection bietet die Möglichkeit, die Struktur von Assemblies, Klassen, Methoden etc. zu inspizieren

```
using System.Reflection;

//...

  Assembly a = Assembly.GetCallingAssembly();
  Type[] types = a.getTypes();

  foreach ( Type t in types ) {

    if ( t.Name == „KlasseDieIchSuche“ ) {
      Methods[] methods = t.getMethods();
      // ...
    }

  }
```

http://www.msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemreflectionassemblyclasstopic.asp