Programmiermethoden in der Mathematik
Wintersemester 03/04

TU Berlin, Institut für Mathematik
Thomas Slawig

Objektorientierte Programmierung – Grundlagen und -Begriffe

Objektorientierung/Objektorientierte Programmierung (OO)

- ist eine spezielle Herangehensweise an Zusammenhänge, Probleme und Lösungsmöglichkeiten im Bereich der Informationswissenschaften
- ist mittlerweile sehr populär und wird als Gegensatz (oder Ablösung) der sog. *prozeduralen* (oder auch *strukturierten*) *Programmierung* gesehen
- kann auch abstrakter als Werkzeug zur Beschreibung von komplexen Strukturen, Zusammenhängen, Problemen und Lösungswegen benutzt werden
- man spricht von der objektorientierten Denkweise, von einem Paradigma (Weltbild, Modell, Muster)
- entstand vor ca. 30 Jahren
- wurde in vielen Programmiersprachen umgesetzt, z.B. *Simula*, *Eiffel* ...
- in viele prozedurale Sprachen wurden objektorientierte Elemente eingebaut z.B. *Smalltalk*, *Objective C*, *C++*, *Delphi* ...

Was ist das Besondere an der OO im Gegensatz zur prozeduralen Programmierung?

- Prozedurale Programmierung
  - hat eine streng gepflegte Trennung zwischen Daten und Operationen/Funktionen (Daten- und Funktionsschicht)
  - entspricht dem für den Menschen gewohnten Denken in Kausalketten (eindeutige Trennung von Ursache und Wirkung)
- Objektorientierte Programmierung
  - sieht Daten und Operationen/Funktionen als Einheit. Motivation dafür ist:
    - Daten alleine haben keinen Sinn, sie werden erst durch die mit ihnen durchgeführten Operationen sinnvoll.
    - Bestimmte Operationen haben für verschiedene Daten auch verschiedene Auswirkungen.
  - entspricht daher eher einem *systemtheoretischen Ansatz*, der die Wirklichkeit als ein System betrachtet, in dem sich alle Teile gegenseitig beinflussen.

Als Vorteile des objektorientierten Ansatzes werden weiter genannt:

- Bessere Abstraktionsmöglichkeit: Modellierung wird stärker vom Lösungs- in den Problem- (oder Ziel-)Bereich verschoben.
- Starke Möglichkeit der hierarchischen Abstraktion:
  - Teile-Ganzes-Beziehung (Bsp.: Ein Auto *hat* eine Farbe).
  - Oberbegriff-Beziehung (Bsp.: Ein Auto *ist* ein Fahrzeug). Diese Art der Abstraktion gibt es nur in der objektorientierten Programmierung.
- Methodische Durchgängigkeit: In allen Phasen der Softwareentwicklung wird mit denselben Konzepten gearbeitet.
- Durch die Einheit von Daten und Operationen steht die Nutzbarmachung durch den Menschen mehr im Vordergrund.
- Evolutionäre Entwicklung wird begünstigt: Eine objektorientiert angelegte Software ist flexibler gegenüber Änderungen und Erweiterungen.

- OO vereinfacht (Software-)Entwicklung in Gruppen (Projektarbeit).
- Höhere Sicherheit und Qualität der Software.

Grundprinzipien und -Begriffe der Objektorientierten Programmierung:

- **Objekt-Klassen-Beziehung:**
  - Eine **Klasse** ist ein abstrakter Oberbegriff für Dinge (Objekte), die eine gemeinsame Struktur und/oder ein gemeinsames Verhalten haben.
  - Ein **Objekt** ist eine zur Laufzeit eines Programms vorhandenes **Exemplar** einer Klasse, für das Speicherplatz zur Verfügung gestellt ist. Man sagt auch: Ein Objekt ist eine **Instanz** einer Klasse (von engl. *instance*: Beispiel, eintretender Fall). Wird ein Exemplar einer Klasse erzeugt, so spricht man von **Instanzierung**.
  - Die Beziehung zwischen Klasse und Exemplar (Instanz) heißt **Objekt-Klassen-, Exemplar-** oder **Instanzbeziehung**.

  Beispiel: Definiert man *Auto* als eine Klasse, so ist mein rotes Auto der Marke ..., das draußen vor der Uni steht, ein Objekt (ein Exemplar, eine Instanz) der Klasse *Auto*.

- **Kapselungsprinzip:**
  Eine Klasse und damit ihre Exemplare werden durch folgende Eigenschaften charakterisiert:
  - **Attribute** (oder **Daten**) beschreiben die Struktur der Objekte einer Klasse: Ihre Bestandteile und die in ihnen enthaltenen Informationen.
  - **Operationen** (oder **Methoden**, Elementfunktionen) beschreiben das Verhalten der Objekte einer Klasse. Operationen können über **Parameter** verfügen.
  - **Zusicherungen** sind Bedingungen, Voraussetzungen und Regeln, die die Objekte erfüllen müssen.
  - **Beziehungen** können zu anderen Klassen gegeben sein.

  Beispiel: Ein Auto kann z.B. die Attribute *Farbe, Marke, Geschwindigkeit* ... haben.
  Als Methoden gibt es z.B. *Auto starten*, *Auto beschleunigen*, *Auto bremsen*, *Auto einparken* ...
  Die Methode *Auto beschleunigen* bekommt als Parameter z.B. die gewünschte Endgeschwindigkeit, also beispielsweise *50 km/h*.
  Eine sinnvolle Zusicherung für das Attribut *Geschwindigkeit* ist, dass sie ≥ 0 ist.
  Die Klasse *Auto* kann eine (noch genauer zu spezifizierende) Beziehung zu einer Klasse *Fahrzeug* haben.

  Klassen fassen Attribute und Methoden zu einer Einheit zusammen. Attribute sind standardmäßig nur über die Methoden der Klasse zugänglich. Dies versteht man unter **Zugriffsbeschränkung** oder **Kapselung**. Attribute und Methoden sind entweder
  - zum Zugriff von außen *freigegeben* (sie sind dann **öffentlich** oder engl. `public`)
  - oder aber vor dem Zugriff *geschützt* (sie sind dann **geschützt**, engl. `protected`, oder sogar **privat**, engl. `private`).

  Beispiel: Der direkte Zugriff auf das Attribut *Geschwindigkeit* eines Objektes der Klasse *Auto* kann z.B. geschützt und nur über die (öffentlichen) Methoden *Auto beschleunigen* und *Auto bremsen* möglich sein.

- **Objektidentität:**
  Ein Objekt ist unabhängig von den konkreten Werten seiner Attribute von anderen Objekten eindeutig zu unterscheiden.

  Beispiel: Selbst wenn ein Auto derselben Marke und Farbe mit derselben Geschwindigkeit fährt wie meins, so ist es eben doch ein anderes Auto.

- **Konstruktoren und Destruktoren:**
  Erzeugen und Zerstören eines Objektes einer Klasse erfolgen mit speziellen Methoden, dem **Konstruktor** bzw. **Destruktor**. Sie stellen den benötigten Speicherplatz zur Verfügung bzw. geben ihn wieder frei. Es kann mehrere Konstruktoren geben, z.B. ohne oder mit gleichzeitiger Initialisierung. Diese unterscheiden sich nur an der Anzahl und Art ihrer Parameter.