Einführung in die Objektorientierte Programmierung

- Elemente des Problemlösens
- Probleme lösen mit dem Computer
- Beschreiben von Objekten
- Beschreiben von Klassen
- Programmieren mit Java-Objekten

Objekte und Klassen

Elemente des Problemlösens

Beispiel: **Kuchen backen**

benötigt werden:

- Zutaten
- "Werkzeuge", wie Backform, Herd, Rührgerät, Uhr
- Rezept

Zutaten: Backmischung, Wasser, Öl

Backform: gefettet und leicht mit Mehl bestäubt
Herd: Ober-/Unterhitze: 225°C; Heissluft: 200°C
Rührgerät: auf höchster Stufe (3)
Uhr: eingestellt auf 45 Minuten

Rezept: Backofen vorheizen, Backform fetten;
Backmischung, Wasser und Öl in eine Schüssel geben, auf höchster Stufe zwei Minuten rühren;
45 Minuten backen, der Kuchen ist fertig, wenn die Oberfläche gleichmäßig goldbraun ist;

Wenn wir das Rezept befolgen, erhalten wir einen optimal gebackenen Kuchen.
Das auf der Backmischung angegebene **Rezept** ist das Endprodukt eines Problemlösungsprozesses.

Objekte und Klassen

Elemente des Problemlösens

Beispiel: **neuen Wasserhahn montieren**

Hier gibt es üblicherweise kein fest vorgegebenes Rezept, an das sich der Heimwerker bzw. der Installateur halten können.

Wir benötigen zunächst eine **Analyse** des Problems, um festzulegen, *was* zu tun ist.

Anschließend legt der **Entwurf** fest, *wie* das Problem gelöst werden soll.

Objekte und Klassen

Elemente des Problemlösens

Objekte sind die Bausteine und Werkzeuge, die zusammenwirken, um das Problem zu lösen.

Ein Objekt besteht aus **Attributen**, die seinen Zustand beschreiben und aus **Aktionen**, die sein Verhalten beschreiben.

Beispiel:

Beschreibung eines **Herd-Objekt**s:

Attribute:

- Größe
- Temperatur
- Hitzequelle (Herdplatte, Backofen)

Aktionen

- Herd an- bzw. ausschalten
- Hitzequelle auswählen (kochen, backen)
- Temperatur einstellen

Objekte und Klassen

Elemente des Problemlösens

Wenn die Objekte festgelegt worden sind, müssen wir einen **Agenten** identifizieren, der das Zusammenwirken der Objekte organisiert, um die Aufgabe zu erfüllen.

Beispiele:

Kuchen backen:

Agent ist der Hausmann bzw. der Bäcker

Wasserhahn montieren:

Agent ist der Heimwerker bzw. der Klempner

Die Objektorientierte Softwareentwicklung betrachtet das Problemlösen aus der Sicht von Objekten.

Die **Analysephase** identifiziert Objekte als die Elemente des Problemlösungsprozesses.

Die **Entwurfsphase** legt ein Rezept fest, das es einem Agenten ermöglicht, die Aktionen der Objekte zu koordinieren.

Objekte und Klassen

Elemente des Problemlösens

zwei Probleme aus dem "richtigen Leben":

1) Sie sitzen spätabends in Ihrem Zimmer mit der Aufgabe, Kapitel 1 eines Java-Lehrbuchs durchzu-arbeiten und eine Zusammenfassung zu schreiben.

Zum Lösen dieses Problems benötigen Sie eine Reihe von **Objekten**:

- Buch
- Licht
- Papier
- Stift

Der **Agent**, der das Licht einschalten, das Buch öffnen und die Zusammenfassung schreiben muss, sind Sie.

2) Eine Fernbedienung macht dem Fernsehzuschauer das Leben leichter. Die Fernbedienung hat eine Tastatur, mit der die Funktionen des Fernsehgeräts gesteuert werden können.

Der Zuschauer ist der **Agent**, der verantwortlich ist für das Einschalten des Geräts, die Auswahl des Kanals und die Lautstärkeregelung.

Objekte und Klassen

Probleme lösen mit dem Computer

Ein mittels Computer durchzuführender Prozess wird entworfen, um einen Informationsfluss zu verarbeiten, bei dem

- Eingabedaten in den Speicher eingelesen werden,
- Berechnungen auf diesen Daten durchgeführt werden und
- die Ergebnisse als Ausgabedaten bereitgestellt werden.

Wenn ein Computer am Problemlösungs-Prozess beteiligt ist, müssen wir uns mit Objekten beschäftigen,

- deren **Attribute** *Daten* sind und
- deren **Aktionen** *Operationen* sind, *die diese Daten verarbeiten*.

Objekte und Klassen

Probleme lösen mit dem Computer

Modellieren von Objekten der realen Welt

Computer-Objekte repräsentieren abstrakte Modelle der realen Welt. Sie isolieren die Daten, die in einer bestimmten Problemsituation relevant sind.

Beispiel:

Akten, die an einer Hochschule über die Studierenden geführt werden

In der realen Welt haben Studierende

- physische Attribute (Geschlecht, Alter, Haarfarbe, Schuhgröße, Kragenweite, ...)
- demographische Informationen (Semesteranschrift, Heimatanschrift)
- hochschulinterne Angaben (Studiengang, Noten)

Beim Eintritt in die Hochschule haben Studierende mit verschiedenen Abteilungen zu tun, die unterschiedliche Daten über die Studierenden speichern und verarbeiten:

- Studienservice (mit Finanzverwaltung)
- Fachbereich
- Prüfungsamt
- Hochschulsport
- Studentenwerk

Objekte und Klassen

Probleme lösen mit dem Computer

Modellieren von Objekten der realen Welt

Jede dieser Abteilungen verarbeitet ganz spezielle Daten. Wir müssen daher verschiedene Modelle eines Studierenden erzeugen.

Die *Finanzverwaltung* benötigt Angaben über

- die Matrikelnummer,
- die Rechnungsanschrift (wegen der Studiengebühren) und
- den aktuellen Kontostand.

Studentenwerk und *Hochschulsport* benötigen Angaben über

- Geschlecht und
- Alter.

Das *Prüfungsamt* verarbeitet Daten über

- den gewählten Studiengang und
- die erzielten Noten.

Objekte und Klassen

Probleme lösen mit dem Computer

Modellieren von Objekten der realen Welt

Zur Beschreibung einer Studierenden an einer Hochschule erzeugen wir eine Reihe von Objekten, deren Attribute die Interaktion zwischen der Studierenden und den verschiedenen Abteilungen der Hochschule modellieren.

Zum Beispiel:

Konto

Attribute: matrikelnr, name, kontostand

Studiengang

Attribute: matrikelnr, hauptfach, durchschnittsnote (gpa)

Wohnheim

Attribute: matrikelnr, geschlecht, alter

Objekte und Klassen

Probleme lösen mit dem Computer

Modellieren von Objekten der realen Welt

Beispiel:

Für die Studierende Mary Winters sind die folgenden Angaben gespeichert:

matrikelnr	59-1420
name	Mary Winters
gpa	3.45
kontostand	-2650.00
hauptfach	Architektur
geschlecht	weiblich
alter	21

Objekte und Klassen

Probleme lösen mit dem Computer

Modellieren von Objekten der realen Welt

Die Angaben über Mary Winters sind folgendermaßen auf die verschiedenen Arten von Objekten verteilt:

Konto

matrikelnr	59-1420
name	Mary Winters
kontostand	-2650.00

Studiengang

matrikelnr	59-1420
hauptfach	Architektur
gpa	3.45

Wohnheim

matrikelnr	59-1420
geschlecht	weiblich
alter	21

Objekte und Klassen

Probleme lösen mit dem Computer

Modellieren von Objekten der realen Welt

Die Objekte müssen neben den *Attributen* auch die *Operationen* modellieren, welche die Interaktion zwischen einem Studierenden und den verschiedenen Abteilungen beschreiben.

Die Finanzverwaltung z. B. führt für jeden Studierenden ein Konto, von dem etwa die Studiengebühren, die Miete für das Zimmer im Wohnheim oder die Kosten für das Mensaessen abgebucht werden (Operation *belastung*).

Es müssen natürlich auch Einzahlungen oder Überweisungen auf das Konto vorgenommen werden, um das Konto auszugleichen (Operation *gutschrift*).

Es werden regelmäßig Kontoauszüge gedruckt (Operation *druckeKontoauszug*).

Die Operation *getKontostand* erlaubt den anderen Abteilungen der Hochschule den lesenden Zugriff auf den aktuellen Kontostand.

Objekte und Klassen

Probleme lösen mit dem Computer

Modellieren von Objekten der realen Welt

Das Objekt *Konto* könnte z. B. die folgenden Operationen besitzen:

belastung:
einen Betrag abbuchen

gutschrift:
einen Betrag einzahlen

getKontostand:
den aktuellen Kontostand erfragen

getMatrikelnr:
die Matrikelnr erfragen

getName:
den Namen erfragen

druckeKontoauszug:
Kontoauszug drucken

Objekte und Klassen

Beschreiben von (Computer-)Objekten

Die Finanzverwaltung legt natürlich nicht nur für Mary Winters ein Konto-Objekt an. Sie führt für jeden Studierenden ein Konto.

Alle diese Konten haben die beschriebenen Attribute und Operationen. Diese definieren eine Schablone, welche die Struktur eines Konto-Objekts festlegt.

Eine solche Objekt-Schablone bezeichnen wir als **Klasse** (seltener auch als **Objekt-Typ**).

Ein Objekt wird als **Exemplar** oder **Instanz** einer Klasse bezeichnet.

Definition:

Eine **Klasse** ist eine Schablone, welche die Attribute und Operationen beschreibt, die jedes Objekt dieser Klasse besitzt.

Objekte und Klassen

Beschreiben von Objekten

Beispiele für Konto-Objekte:

Konto-Objekt *mWinters*:

matrikelnr	59-1420
name	Mary Winters
kontostand	-2650.00

Konto-Objekt *dHuey*:

matrikelnr	99-1234
name	David Huey
kontostand	0.00

Objekte und Klassen

Beschreiben von Klassen

Eine Klassenbeschreibung besteht aus drei Teilen:

- Überschrift mit Name und kurzer Beschreibung des Zwecks der Klasse

- Beschreibung der Attribute

- Beschreibung der Operationen

Objekte und Klassen

Beschreiben von Klassen

Beschreibung der Klasse **StudentenKonto**

Ein StudentenKonto verwaltet die Konteninformationen für einen Studenten.

Attribute:

```
matrikelnr      // Matrikelnr. in der Form ##-#####
name            // Name des Studenten
kontostand      // aktueller Kontostand
```

Operationen:

```
// Buchung durchführen; Kontostand aktualisieren
belastung
gutschrift

// aktuelle Werte der Attribute zurückliefern
getMatrikelnr
getName
getKontostand

// Kontoauszug drucken
druckeKontoauszug
```

Objekte und Klassen

Beschreiben von Klassen

Beschreiben von Operationen

Operationen können haben:

- Argumente
- Rückgabewert

Argumente einer Operation sind die Werte, die ein Agent zur Verfügung stellt, wenn er die Operation verwendet.

Beispiel:

Operation	*Argument*
belastung	betrag
gutschrift	betrag

Der **Rückgabewert** einer Operation dient dazu, Informationen an einen Agenten zu übergeben.

Beispiel:

Operation	*Rückgabewert*
getKontostand	aktueller Kontostand

Objekte und Klassen

Beschreiben von Operationen

Beispiel:

Operationen:

belastung
Der Betrag der neuen Belastung wird als Argument übergeben und der Wert wird vom aktuellen Kontostand abgezogen.
Es gibt keinen Rückgabewert.

neuer Kontostand = aktueller Kontostand - betrag

gutschrift
Der Betrag der neuen Gutschrift wird als Argument übergeben und der Wert wird auf den aktuellen Kontostand addiert.
Es gibt keinen Rückgabewert.

neuer Kontostand = aktueller Kontostand + betrag

getMatrikelnr
Gibt die Matrikelnummer des Studenten zurück.

getName
Gibt den Namen des Studenten zurück.

getKontostand
Gibt den aktuellen Kontostand des StudentenKonto-Objekts zurück.

druckeKontoauszug
Liefert eine formatierte Ausgabe von Matrikelnummer, Name und Kontostand

Objekte und Klassen

Beschreiben von Klassen

Initialisieren von Objekt-Attributen

Klassen haben eine Operation namens **Konstruktor**, deren Aufgabe es ist, die Attribute eines Objekts zu initialisieren.

Der Konstruktor kann Argumente haben; diese legen die Initialisierungswerte für die Attribute fest.

Der Konstruktor wird aufgerufen, wenn ein Objekt erzeugt wird. Er kopiert die Werte seiner Argumente in die zugehörigen Attribute.

Der Konstruktor hat den gleichen Namen wie die Klasse.

Beispiel:

Operation:

StudentenKonto

Der Konstruktor hat als Argumente die Initialisierungswerte für die Attribute matrikelnr, name und kontostand.

Objekte und Klassen

Beschreibung der Klasse **StudentenKonto**

Ein StudentenKonto verwaltet die Konteninformationen für einen Studenten.

Attribute:

matrikelnr	// Matrikelnr. in der Form ##-####
name	// Name des Studenten
kontostand	// aktueller Kontostand

Operationen:

StudentenKonto
Der Konstruktor hat als Argumente die Initialisierungswerte für die Attribute matrikelnr, name und kontostand.

belastung
Der Betrag der neuen Belastung wird als Argument übergeben und der Wert wird vom aktuellen Kontostand abgezogen.
Es gibt keinen Rückgabewert.
neuer Kontostand = aktueller Kontostand - betrag

gutschrift
Der Betrag der neuen Gutschrift wird als Argument übergeben und der Wert wird auf den aktuellen Kontostand addiert.
Es gibt keinen Rückgabewert.
neuer Kontostand = aktueller Kontostand + betrag

getMatrikelnr
Gibt die Matrikelnummer des Studenten zurück.

getName
Gibt den Namen des Studenten zurück.

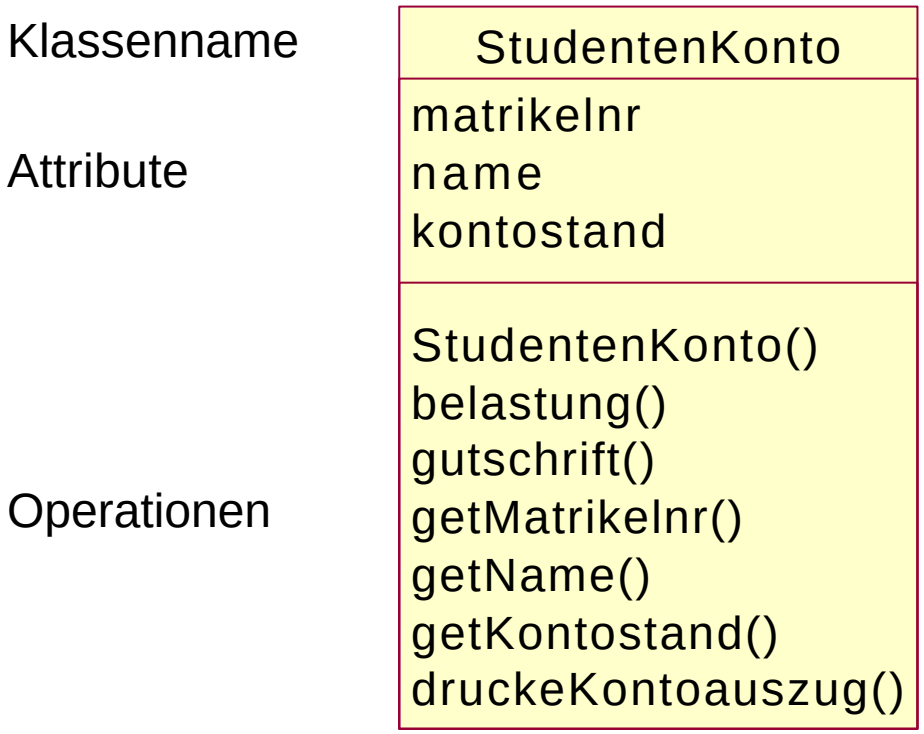
getKontostand
Gibt den aktuellen Kontostand des StudentenKonto-Objekts zurück.

druckeKontoauszug
Liefert eine formatierte Ausgabe von Matrikelnummer, Name und Kontostand

Beschreiben von Klassen

Graphische Beschreibung von Klassen

Wir verwenden die **Unified Modeling Language** (**UML**) zur graphischen Repräsentation von Klassen.



Objekte und Klassen

Programmieren mit Java-Objekten

Computer-Programme sind ein mächtiges Hilfsmittel zur Lösung von Problemen.

Die Entwicklung von Programmen beginnt mit der Analyse des Problems und mit der Erzeugung einer Folge von einzelnen Schritten, die zu einer Lösung führen.

Diese Folge von Schritten wird **Algorithmus** genannt.

Wir implementieren die einzelnen Schritte mit Java-Anweisungen.

Definition:

Ein **Algorithmus** ist eine Folge von Anweisungen, die in endlicher Zeit zur Lösung eines Problems führt.

Objekte und Klassen

Programmieren mit Java-Objekten

Objekte deklarieren, erzeugen und verwenden

Beispiel:

Deklaration:

```
StudentenKonto fKadelle, kToffel;
```

Erzeugung:

```
fKadelle = new StudentenKonto("23-9876",
             "Fritz Kadelle", 0.0);
kToffel  = new StudentenKonto("67-5634",
             "Karl Toffel", 3600.0);
```

Verwendung:

```
fKadelle.gutschrift(1000.0);
System.out.println(fKadelle.getKontostand());

kToffel.belastung(79.95);
kToffel.druckeKontoauszug();
```

Objekte und Klassen

Beschreiben von Operationen

Wir wollen die Operationen einer Klasse formaler beschreiben. Zusätzlich zur bisherigen Beschreibung geben wir in Java-Notation an:

Name, **Rückgabetyp**, **Argumentliste**

Operationen:

Operationsname
Beschreibung der Aktion, der Argumente und des Rückgabewerts
rückgabeTyp operationsName (argumentListe)

Beispiel:

Operationen:

belastung
Der Betrag der neuen Belastung wird als Argument übergeben und der Wert wird vom aktuellen Kontostand abgezogen.
void belastung(double betrag)

getMatrikelnr
Gibt die Matrikelnummer des Studenten zurück.
String getMatrikelnr()

druckeKontoauszug
Liefert eine formatierte Ausgabe von Matrikelnummer, Name und Kontostand
void druckeKontoauszug()

Objekte und Klassen

Beschreiben von Klassen

Beschreibung der Klasse **StudentenKonto**

Ein StudentenKonto verwaltet die Konteninformationen
für einen Studenten.

Attribute:

```
String matrikelnr      // Matrikelnr. in der Form ##-#####
String name            // Name des Studenten
double kontostand      // aktueller Kontostand
```

Operationen:

```
// Konstruktor
StudentenKonto(String mnr, String name, double betrag)

// Buchung durchführen; Kontostand aktualisieren
void belastung(double betrag)
void gutschrift(double betrag)

// aktuelle Werte der Attribute zurückliefern
String getMatrikelnr()
String getName()
double getKontostand()

// Kontoauszug drucken
void druckeKontoauszug()
```

Objekte und Klassen

Beschreiben von Klassen

Graphische Beschreibung von Klassen

Argumente und Rückgabetyp der Operationen werden in die UML-Beschreibung mit aufgenommen.

StudentenKonto
matrikelnr : String name : String kontostand : double
StudentenKonto(mnr : String, name : String, betrag : double) belastung(betrag : double) : void gutschrift(betrag : double) : void getMatrikelnr() : String getName() : String getKontostand() : double druckeKontoauszug() : void

UML-Format

```
objName : ObjTyp
funktion(argumente) : ObjTyp
```

Java-Format

```
ObjTyp objName
ObjTyp funktion(argumente)
```