Programmieren I

Dr. Klaus Höppner

Hochschule Darmstadt – Wintersemester 2009/2010

Zielsetzung der Vorlesung

- Ziel ist die Einführung in das Konzept der
  Objektorientierten Programmierung
  unter Verwendung von Java
- Vermittlung von Fähigkeiten wie:
  Strukturierung von Problemen,
  Abstraktionsvermögen,
  algorithmisches Denken

Literatur

- Christian Ullenboom, *Java ist auch eine Insel*, 1475 S., Galileo Computing, 8. Aufl. 2009, ISBN 3836213710, 49,90 EUR
- Cornelia Heinisch, F. Müller-Hoffmann, J. Goll, *Java als erste Programmiersprache*, 1235 S., Teubner, 5. Aufl. 2007, ISBN 3835101471, 35,90 EUR
- Guido Krüger, T. Stark, *Handbuch der Java-Programmierung*, 1356 S., Addison-Wesley, 6. Aufl. 2009, ISBN 3827328748, 49,80 EUR
- Kathy Sierra, Bert Bates , *Java von Kopf bis Fuß*, 688 S, O'Reilly, 1. Aufl. 2006, ISBN 3897214482, 49,90 EUR
- Patrick Kiwitter, *Eclipse in der Java-Entwicklung*, 333 S., Addison-Wesley, 2008, ISBN 3827324904, 34,95 EUR

Definition

- Programmieren bezeichnet die Tätigkeit, Computerprogramme (Software) zu erstellen.
- Im weiteren Sinne versteht man dabei alle Tätigkeiten, die mit dieser Programmerstellung verbunden sind, insbesondere den konzeptionellen Entwurf.
- Im engeren Sinne bezeichnet Programmieren lediglich das Umsetzen dieses konzeptionellen, abstrakten Entwurfes in konkreten Programmcode.

Definition (Forts.)

- Bei kleinen Softwareprojekten sind die Tätigkeit des Entwurfs und der Implementierung häufig nicht getrennt, d. h. das Programm entsteht in enger Wechselwirkung mit dem Entwurf
- In großen Softwareprojekten sind die einzelnen Tätigkeiten personell (und teilweise auch örtlich) getrennt.
  Hier ist die Aufgabe des Programmierers, die durch den Entwurf beschriebene Funktionalität und Wechselwirkung mit anderen Komponenten als Programmcode zu implementieren.

Teilbereiche von Softwareprojekten

- Problemanalyse und Spezifikation der Funktionalität zusammen mit dem Auftraggeber
- Erstellung eines Pflichtenheftes zusammen mit dem Auftraggeber
- Entwurf
- insbesondere Entwurf der Schnittstellen der einzelnen Komponenten
- Implementierung (eigentliches Programmieren)
- Qualitätssicherung/Fehlersuche
- Projektmanagement
- Dokumentation (intern und extern)

Teilbereiche von Softwareprojekten (Forts.)

- Programmieren ist eine *kreative* Tätigkeit
- Programmieren setzt *analytisches* und *logisches* Denken voraus
- Programmieren erfordert
  - Analyse eines gegebenen (Teil-)Problems
  - Entwurf eines **Algorithmus**
  - Implementierung des Algorithmus in einer Programmiersprache

Algorithmus: Definition

- Ein Algorithmus ist eine *wohldefinierte endliche* Methode oder Prozedur zur Lösung eines Problems.
- Typischerweise wird ein Algorithmus durch eine *Folge von Anweisungen* beschrieben, die nacheinander ausgeführt und oft in festgelegter Weise wiederholt werden.
- Algorithmen werden meistens als Computerprogramm implementiert. Möglich ist aber auch die Implementierung als elektronischer Schaltkreis oder die manuelle Ausführung durch Menschen.

Quelle: Wikipedia

Algorithmen: Grundprinzipien

- Ein Algorithmus basiert auf zwei Grundelementen:
  1. den Rechenanweisungen
  2. und bedingten Sprüngen.

  Ein bedingter Sprung zeigt an, an welcher Stelle im Algorithmus fortgefahren werden soll, wenn eine bestimmte Bedingung erfüllt ist, bzw. wenn diese nicht erfüllt ist.
- Bedingungen für einen Algorithmus:
  1. Das Verfahren muss in einem endlichen Text beschreibbar sein
  2. Jeder Schritt muss auch tatsächlich ausführbar sein
  3. Der Ablauf ist zu jedem Zeitpunkt eindeutig definiert
  4. Das Verfahren muss in endlich vielen Schritten zum Ende gelangen.

Beispiel 1: Mathematik

Algorithmus von Euklid

Aufgabe: Ermittlung des größten gemeinsamen Teilers (ggT) zweier natürlicher Zahlen A und B

Algorithmus: Es gilt folgende Handlungsanweisung:

1. Vertausche A und B so, dass $A \geq B$
2. Setze $A = A - B$
3. Falls $A \neq 0$ gilt, fahre mit Schritt 1 fort, sonst beende den Algorithmus

Der Endwert in B ist der größte gemeinsame Teiler.

Schritt	A	B
	18	30
1	30	18
2	12	18
3	18	12
4	6	12
5	12	6
6	6	6
7	0	6

Algorithmen: Geschichtliches

Das Wort Algorithmus ist eine Abwandlung oder Verballhornung des Namens al-Khwarizmi (ca. 783–ca. 850), des Autors des Buchs Kitab al-jabr w'al-muqabala (825, Regeln zur Wiederherstellung und Reduktion), durch das die Algebra im Westen verbreitet wurde. Die lateinische Fassung beginnt mit: »Dixit Algorithmi...«, womit der Autor gemeint war.

Formal wurden Algorithmen durch Alan Turing beschrieben. Dieser führte die so genannte *Turing-Maschine* als ein abstraktes Modell eines Computers ein und zeigte, dass alle *wohldefinierten Prozeduren* durch eine Turing-Maschine emuliert werden können.

Beispiel 2: Sandkuchen

250 g Butter
250 g Zucker
1 Prise Salz
5 Eier
250 g Speisestärke
4 cl Rum
abgeriebene Schale einer Zitrone

Die Butter in einer Schüssel cremig rühren; nach und nach den Zucker und das Salz zugeben. Eier trennen. Abwechselnd das Eigelb mit der Stärke hineinrühren, den Rum und Zitronenschale untermischen. Zuletzt das zu steifem Schnee geschlagene Eiweiß unterziehen. Den Teig in eine mit Butter gefettete und Bröseln ausgestreute Kranzform geben.

Ein Kochrezept enthält:

Datenbereich: Zutatenliste

Datentypen:

- Anzahl *(5 Eier)*
- Masse *(250 g Zucker)*
- Volumen *(4 cl Rum)*

Programmbereich: Handlungsanweisungen

Typische Handlungsanweisungen:

Die Handlungsanweisung eines Kochrezeptes enthält Elemente, die man auch in den meisten Programmiersprachen findet:

Zuweisungen: *Den Rum zum Teig hinzugeben*
$\text{Teig} \leftarrow \text{Teig} + \text{Rum}$

Bedingte Anweisungen: ***Falls** der Teig zerbröselt, **dann** geben Sie etwas Wasser hinzu.*

Schleifen: *Butter **solange** verrühren, **bis** sie schaumig ist*

Funktionen: *Eier trennen*

Funktionen

Funktionen erledigen eine Teilaufgabe und bestehen aus einem Stück Programmcode, der bei Aufruf der Funktion ausgeführt wird.

Hierbei unterscheidet man:

- Funktionen, die im Progamm selber implementiert werden
  Beispiel: Im Glossar eines Kochbuchs werden seltene Prozeduren beim Kochen erklärt.
- Funktionen, deren Vorhandensein voraus gesetzt wird (Bibliotheks-Funktionen).
  Kochbuch-Beispiel: Es wird voraus gesetzt, dass der Koch die Bedeutung der Prozedur *Eier trennen* kennt.
  Software-Beispiel: Standard-Bibliotheken für Ein-/Ausgabefunktionen

Was ist eine Programmiersprache?

Eine Programmiersprache ist eine *formale Sprache* zur Darstellung von Computerprogrammen. Sie vermittelt dem Computer den vom Menschen implementierten Algorithmus und die dabei beteiligten Daten.

Besondere Ausprägungen:

- Maschinensprache
- Hochsprachen, dabei insbesondere
  - Prozedurale Sprachen
  - Objektorientierte Sprachen

Maschinensprache

- Unter Maschinensprache versteht man die Sprache, die direkt auf einer bestimmten Rechnerarchitektur (z.B. i*x*86, PowerPC, Alpha, Motorola 68000) ausführbar ist.
- Für den Menschen nicht lesbar
- Typische Befehle:
  - Lade den Inhalt von Speicherzelle *a* in Register *x*
  - Addiere zu Register *x* den Inhalt von Speicherzelle *b*
  - Schreibe den Inhalt von Register *x* in Speicherzelle *c*
  - Falls das Register *y* den Wert 0 hat, setze die Abarbeitung des Programmcodes an Stelle *z* fort

Hochsprachen

Menschen schreiben ein Programm in einer Hochsprache in Form eines Quelltextes.

Damit der Computer dieses Programm versteht, muss es in Maschinensprache umgewandelt werden.
Dies kann geschehen mit:

- Interpreter
  Der Quelltext wird während der Ausführung *interpretiert*, also zur Laufzeit in Maschinencode umgewandelt. Dies geschieht bei jeder Auführung neu.
- Compiler
  Der Quelltext wird einmalig vor der Ausführung *kompiliert*, d. h. in Maschinensprache übersetzt (und zusammen mit den verwendeten Bibliotheks-Funktionen *gelinkt*).

Hochsprachen (Forts.)

- Ausführen von Bytecode in einer virtuellen Maschine
  Einige Programmiersprachen, insbesondere Java, verwenden zwar Compiler, die allerdings nicht direkt Maschinencode sondern einen Zwischencode erzeugen, den so genannten Bytecode.

  Dieser Bytecode wird dann in der Laufzeitumgebung (Virtuelle Maschine) ausgeführt.

  Ein wesentlicher Vorteil des Bytecodes besteht darin, dass dieser in der Regel von der Maschine und Plattform unabhängig ist. Nachteil liegt in einem geringfügigen Geschwindigkeitsnachteil gegenüber der direkten Ausführung von Maschinencode, da im Fall der Virtuellen Maschine dieser zur Laufzeit aus dem Bytecode erzeugt werden muss.

Kleine Zeittafel von Hochsprachen

Jahr	Sprache
1954	FORTRAN
1960	COBOL
1965	BASIC
1971	Pascal
1972	C
1983	C++
1987	Perl
1991	Python
1995	Java
1997	PHP

- Von Programmierern wird erwartet, dass sie prinzipiell in der Lage sind, die erworbenen Kenntnisse über das *Konzept des Programmierens* in *beliebigen* Programmiersprachen umzusetzen.
- Heute weit verbreitete Sprachen können in der Zukunft von neuen Sprachen abgelöst werden (s. neue Entwicklungen wie C#, Ruby).
- Es kann vorkommen, dass Programmierer zu einem Projekt stoßen, in dem alte Software gepflegt und weiter entwickelt werden muss, die in veralteten Sprachen geschrieben ist (z. B. COBOL, FORTRAN).

OOP: Motivation

In prozeduralen Sprachen sind die Daten und Funktionen (innerhalb ihres jeweiligen Geltungsbereiches) *öffentlich*, d. h. innerhalb des Programms kann frei auf die Daten und Funktionen zugegriffen werden.

Dies setzt bei der Programmierung besondere Sorgfalt voraus.

Beispiel: Geldautomat
Ein Geldautomat verwaltet die Kontostände der Kunden und zahlt Geld aus.
Hier ist der Programmierer selbst dafür verantwortlich, dass nur dann Geld ausgezahlt wird, wenn dieser Betrag gleichzeitig dem Konto des Kunden belastet wird (und umgekehrt).

OOP: Grundprinzip

- Objektorientierte Programmierung (OOP) ist eine Methode zur Strukturierung von Programmen, bei der Daten und Programmlogik eine Einheit bilden.
- Objektorientierte Sprachen nutzen als Organisationsstruktur *Klassen*
- Klassen beschreiben die gemeinsamen Eigenschaften *(Attribute)* und Fähigkeiten *(Methoden)* von gleichartigen *Objekten*.
- Ein einzelnes Objekt, das durch die Klasse *K* beschrieben wird, wird häufig auch als *Instanz* der Klasse *K* bezeichnet.

Beispiel

- Klasse: Mensch
- gemeinsame Eigenschaften: Haarfarbe, Augenfarbe, Hautfarbe, Größe, Gewicht, . . .
- gemeinsame Fähigkeiten: hören, riechen, sehen, sprechen, . . .

Hans Müller ist eine Instanz der Klasse Mensch mit Haarfarbe *blond*, Augenfarbe *blau*, Hautfarbe *weiß*.
Auch wenn alle Menschen Instanzen der gleichen Klasse Mensch sind, so sind sie doch eindeutig unterscheidbar.

Komplexes Beispiel

- Klasse: Geldautomat
- Daten:
  - Geldbestand im Automaten
  - Liste der Kontostände aller Kunden
- Methoden:
  - gebe Geld aus
  - belaste ein Konto
  - zeige Kontostand
  - zahle Geld vom Konto aus

Private Daten und Methoden

Als *privat* deklariert sind:

- Geldbestand
- Liste der Kontostände
- gebe Geld aus
- belaste Konto

Diese Daten und Methoden sind nur *innerhalb* der Klasse zugänglich.
Dem Kunden bleibt z. B. der Geldbestand im Automaten verborgen. Genauso ist es nicht möglich, die Methode *gebe Geld aus* direkt aufzurufen (und so eine Auszahlung zu erreichen, ohne dass diese einem Konto belastet wird).

Öffentliche Methoden

Als *öffentlich* deklariert sind:

- zeige Kontostand
- zahle Geld vom Konto aus
  Implementation:
  - Falls Geldbestand ausreichend und Konto gedeckt:
    - gebe Geld aus
    - belaste Konto

Hierdurch ist sicher gestellt, dass Geld nur dann ausgezahlt wird, wenn das entsprechende Konto belastet wird (und umgekehrt).

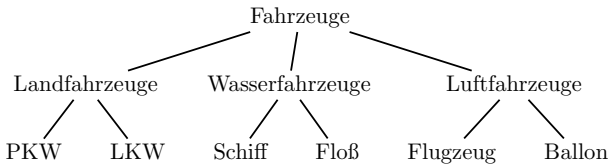
OOP: Grundlegende Eigenschaften

Abstraktion Jedes Objekt „kommuniziert“ mit andern Objekten durch die in der Klassendefinition definierten Schnittstellen. Die eigentliche Implementation der Methoden bleibt verborgen.

Kapselung Die Kommunikation über öffentliche Schnittstellen sorgt dafür, dass der interne Zustand der Objekte nicht in unerwarteter Weise geändert werden kann.

OOP: Grundlegende Eigenschaften

Vererbung Klassen können von anderen Klassen erben.
Dies bedeutet, dass eine neue Klasse die Eigenschaften und Methoden der Ursprungsklasse *erbt*, d. h. übernimmt, ohne dass sie noch einmal neu implementiert werden müssen.
In der neuen Klasse können dann neue Methoden und Eigenschaften definiert werden, weiterhin können z. B. Methoden der Ursprungsklasse überschrieben oder erweitert werden.
Bei der Vererbung entsteht in der Regel aus einer allgemeinen Klasse eine spezialisiertere Klasse



Die Klasse *Fahrzeug* besitzt die Eigenschaft *Geschwindigkeit* und die Methode *(horizontal) bewegen*.
Die Klasse *Luftfahrzeug* besitzt zusätzlich die Eigenschaft *Höhe* und die Methoden *steigen* und *sinken*.