Modul 226a

Dokumentation

Inhaltsverzeichnis

[1 Teil A - Grundlagen objektorientiertes Design 2](#_Toc525818366)

[1.1 Begriffe 2](#_Toc525818367)

[1.2 Anleitungen 4](#_Toc525818368)

[1.3 Beziehungen und Kardinalitäten 4](#_Toc525818369)

[1.4 Codebeispiel 5](#_Toc525818370)

[1.5 Schleifen und Verzweigungen 6](#_Toc525818371)

# Teil A - Grundlagen objektorientiertes Design

## Begriffe

**Main (-) - Methode**: Main (-) {…} dient als Einstiegspunkt für das Java Programm. Da es nur einen Einstieg geben kann, darf diese Methode nur einmal im ganzen Projekt vorkommen.

**Klasse**: In einer Klasse sind Methoden und Eigenschaften eines Objektes definiert. Die Klasse dient als Vorlage, aus der dann beliebig viele Objekte erzeugt werden können.

**Klassen und Datentypen**: Klassen sind Datentypen. Eine Klasse beschreibt, wie sich Objekte verhalten und welche Eigentschaften sie haben.

**Methode**: Eine Methode ist im Grunde genommen eine Funktion. Sie sind Programmteile,

welche bestimmte Teilaufgaben lösen. Es wird gesagt, was jetzt genau passieren sollte.

**Instanz**: Ein "Alias" oder Variablenname für eine Klasse. Klassen können mehrere

Instanzen haben.

// Regisseur **Maersu** = new Regisseur ( );

// Regisseur **Neemo** = new Regisseur ( );

// Datentyp **Variable** = new Methode ( );

**Parameter**: Ein Parameter ist ein Übergabewert. Bei der Deklaration einer Methode kann man so einen Wert "verlangen". Beim Ansprechen der Methode muss daher ein Parameter gesetzt werden:

String aussage = "Beispiel Text";

public void setAussage (String neuerText) {

aussage = neuerText;

}

Regisseur.setAussage("Mit dem wird die Aussage ersetzt.")

OUTPUT: Mit dem wird die Aussage ersetzt.

**Objekt**: Objekte sind konkrete Instanzen von Klassen. Sie haben bestimmte Eigenschaften, definierte Fähigkeiteung und ein bestimmtes Verhalten.

**Konstruktor**: Ein Konstruktor enthält immer den gleichen Namen wie die Klasse.

public class Aussage {

public Aussage() {

aussage = "public Aussage ist ein Konstruktor."

}

}

**Beziehungen**: Zwei Objekte müssen sich kennen, damit sie mit eineinander kommunizieren können. Man unterscheidet zwischen 3 Beziehungsarten:

**Assoziation**: Klasse A kennt Klasse B. Man

**Aggregation**: Klasse B enthält 1 oder mehrere Objekte der Klasse A

**Komposition**: Klasse B muss mindestens 1 Objekt der Klasse A haben.

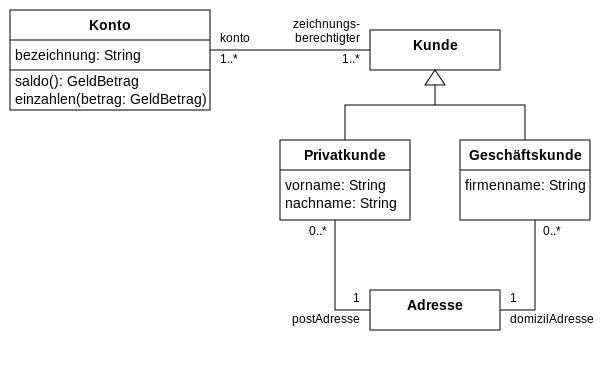
**UML (Unified Modeling Language)**: In objektorientierten Designs (OOD) werden UML's verwendet um eine Übersicht zu erhalten. UML kennt zahlreiche normierte Diagramme zur Darstellung statischer und dynamischer Beziehungen zwischen Klassen und Objekten. In einem ***Klassendiagramm*** werden alle Klassen aufgelistet, und welche Beziehung diese zu einander haben. Dasselbe gibt es auch für Objekte: das **Objektdiagramm**.

Abbildung 1 // Klassendiagramm

Arten der Beziehung:

1 // genau eine  
0,1 // konditionell, keine oder eine   
\* // multiple, keine Einschränkung  
0.. // multiple  
1.. // mindestens eine

## Anleitungen

**Neues Projekt mit Paket erstellen**:

Neues Java-Projekt **//** Projektname eingeben **//** Ansonsten keine Angaben

**Neue Klasse hinzufügen**:

Im Projekt: New > Class **//** Package eingeben **//** Name eingeben **//** Pro Projekt nur 1 Main!

## Beziehungen und Kardinalitäten

SEITE 34 - Assoziation, Aggregation, Komposition

## Codebeispiel

**Hello World**

System.out.println("Hello World!");

**Klasse Regisseur mit zwei Methoden**

public class Regisseur {

public void go() {

System.out.println("Klappe, die erste, Action!");

}

public void stop() {

System.out.println("Alles im Kasten, Szene beendet!");

}

} // Klasse Regisseur mit Methode *go* und *stop*

**Neuer Regisseur (Variablenname) + Ausführen der Methoden (Welche in der Klasse sind)**:

Regisseur Planissimo = new Regisseur(); // Dies kann man mit jeder Klasse machen.

Planissimo.go()

Planissimo.stop() // Ausführen Methode go und stop

## Schleifen und Verzweigungen

**Bedinung |  Auszuführender Code**

*int zahl = 3;*

**while** (zahl < 20) **{**

System.out.println("Aktuelle Zahl ist: " + zahl);

zahl++;

**}**

*int zahl = 4;*

**do {**

System.out.println("Aktuelle Zahl ist: " + zahl);

**}** **while** (zahl < 20);

**for** (int i = 0; i < 5; i++) **{**

System.out.println("Dies ist eine Schleife die 5 Mal ausgeführt wird");

**}**

*int[] vieleZahlen = { 1, 2, 4, 6, 35, 935, 1337 };*

**for** (int element : vieleZahlen) **{**

System.out.println("Alle Zahlen: " + element); **}**

*int zahl = 6;*

**if** (zahl < 5) **{**

System.out.println("Zahl ist kleiner als 5.");  
**}** **else if** (zahl > 5) **{**

System.out.println("Zahl ist grösser als 5.");  
**}** **else {**

System.out.println("Zahl ist 5.");  
**}**

int Ausgabe = ()

*int zahl = 7;*

**Switch** (zahl) **{**

case 6:

System.out.println("Die Zahl ist 6");

break;

case 7:

System.out.println("Die Zahl ist 7");

break;

default:

System.out.println("Falls nichts zutrifft kommt der Default");

**}**

# TEIL B - Sprachliche Grundlagen von Java

**Datentypen**:

byte **//** Ganzzahl (8bit)

short **//** Ganzzahl (16bit)

int **//** Ganzzahl (32bit) int Zahl = 17;

long **//** Ganzzahl (64bit) long Nummer = 2348739271;

float **//** Fliesskommazahl (32bit)

double **//** Fliesskommazahl (64bit) double pi = 3.14159;

boolean **//** true/false

char **//** einzelnes Zeichenchar zeichenP = "P";

String **//** Zeichenkette String wort = "Hallo"; Additionen mit Strings möglich

**Array** **//** Sammlung von Objekten

int [ ] vieleZahlen = { 1, 2, 4, 6, 35, 935, 1337 };

vieleZahlen = new int [ 420 ];

**Collections** **//** Collections sind dynamische Arrays (ArrayList)

ArrayList<Integer> vieleZahlen = new ArrayList<Integer>();

vieleZahlen.add(newInteger(1337)); // fügt 1337 in den Array vieleZahlen hinzu

Integer wert = vieleZahlen.get(3) // verlangt den vierten Wert im Array

System.out.println(zahlen.count()); // zählt alle Elemente im Array

Folgendes überprüft die Länge eines Arrays:

**for** (sohrt index = 0; index < vieleZahlen.length; index++) **{**

vieleZahlen[index] = index;

**}**

**Konstanten** **//** Konstanten sind Variablen, deren Wert nicht mehr geändert werden können. Folgendermassen definiert man eine konstante Variable:

final String unsereNamen = "Marcel und Neemo";

**Wrapperklassen**

Jede Wrapperklasse hat einen Konstruktor, welcher einen primitiven Datentyp als Parameter verlangt.

|  |  |
| --- | --- |
| Primitiver Datentyp | Wrapper-Klasse |
| byte | Byte |
| short | Short |
| int | Integer |
| long | Long |
| boolean | Boolean |
| char | Character |
| float | Float |
| double | Double |
| void | Void |

**Operationszeichen & Operatoren**:

== **//** gleich

!= **//** ungleich

< **//** kleiner als

<= **//** kleiner oder gleich

> **//** grösser als

>= **//** grösser oder gleich

! **//** nicht

&& **//** UND

| | **//** ODER