

Einführung in die objektorientierte Programmierung mit Java

Katja Wegner

katja.wegner@eml-r.villa-bosch.de

Überblick

⇒ Allgemein:

- Theorie
- Programmieren

⇒ Grundlagen:

- Programmaufbau
- Datentypen
- Zuweisungen
- Bedingungen
- Schleifen

Java

1. Programmiersprache
2. Insel (zweitgrößte von Indonesien)
3. Umgangssprachlich für eine Tasse Kaffee in den USA



SDK + API

⇒ **SDK** - Software Development Kit

- Download: <http://java.sun.com>

⇒ **API** - Application Programming Interface

- vollständige Dokumentation aller Klassen und Methoden

Java - Eigenschaften

+

-

einfach und robust

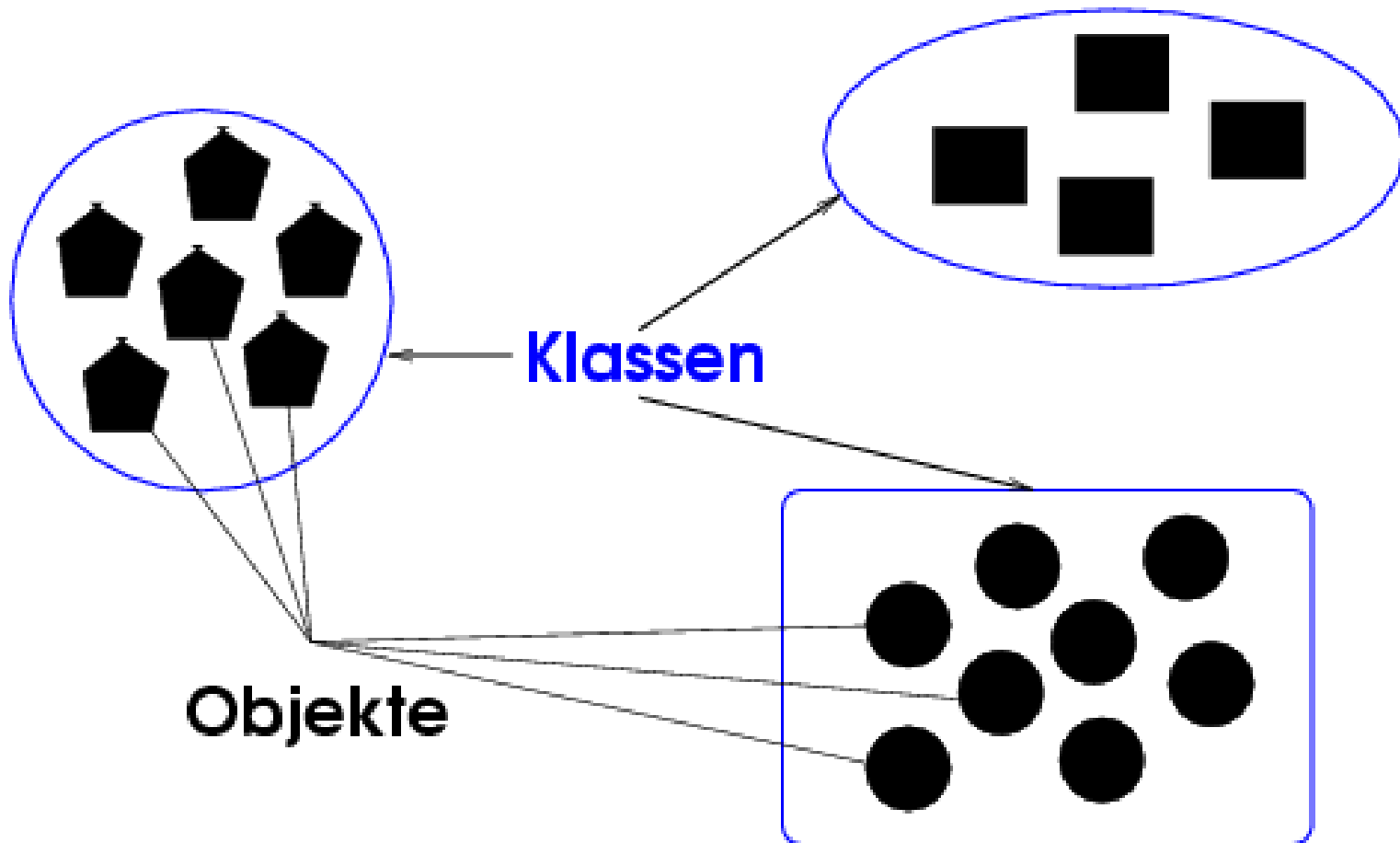
Performance

- keine Zeiger
- keine Speicherverwaltung
- kein Überladen von Operatoren

plattformunabhängig

objektorientiert

Objektorientiert



Programmdefinition

⇒ jedes **Programm** = **Objekt** einer **Klasse**

- Schreiben eines Programms = Definieren einer Klasse von Objekten
- Programmaufruf = Erzeugen eines Objektes einer Klasse

⇒ Klasse besitzt **Attribute** und **Methoden**

⇒ **Beispiel:** Klasse Viereck

- Attribute: Eckpunkte
- Methoden: Transformationen der Eckpunkte (Rotieren, Verschieben, ...)

Attribute

- Name einer Variablen, dem im Programm ein Wert zugeordnet wird
- Zugriff über den Namen auf den Wert der Variablen
- Zugeordnete Werte können während des Programmablauf variieren
- Datentyp schränkt Wertebereich des Attributs ein
- Datentyp bei Deklaration der Variablen festgelegt

Methoden

- Zusammenfassung von Deklarationen und Anweisungen
- Besitzt einen Namen
- Aufruf der Methode im Programm über diesen Namen
- Ersetzt Funktionen, Prozeduren und Unterprogrammen von anderen Programmiersprachen

Klassendefinition

```
public class HelloInfFem05
{
    public static void main(String[] args)
    {
        String text = "Willkommen_auf_der_InfFem05";
        System.out.println(text);
    }
}
```

- **Klasse:** HelloInfFem05
- **Methode:** main()
- **Variable:** text
- **Ausgabeweisung:** System.out.println()

Tipps und Tricks

- ⇒ Zeichengenau eingeben
- ⇒ Groß- und Kleinschreibung beachten
- ⇒ verschiedene Klammern: `()`, `{}`, `[]`
- ⇒ Semikolon `;` am Ende jeder Anweisung
- ⇒ Dateiname = Klassenname + Endung **.java**

Übersetzen und Ausführen

⇒ **javac** = Java-Compiler

- übersetzt Java Quelltext-Dateien (DateiName.java)
- erzeugt Byte-Code-Dateien (DateiName.class)
- Aufruf mit Dateiendung

⇒ **java** = Java-Interpreter

- führt Java-Byte-Code aus
- Aufruf ohne Dateiendung

Linux + Java

- ⇒ **cd** - Verzeichnis wechseln
- ⇒ **mkdir** - Verzeichnis erstellen
- ⇒ **kedit** - Editor
- ⇒ **vim** - Editor

- ⇒ **javac HelloInfFem05.java** - Datei übersetzen
- ⇒ **java HelloInfFem05** - Programm ausführen

Variablen und Konstanten

⇒ Variablen:

1. [modifier] datentyp einName [= wert];
2. [modifier] datentyp einName;
 einName = wert;

⇒ Konstanten:

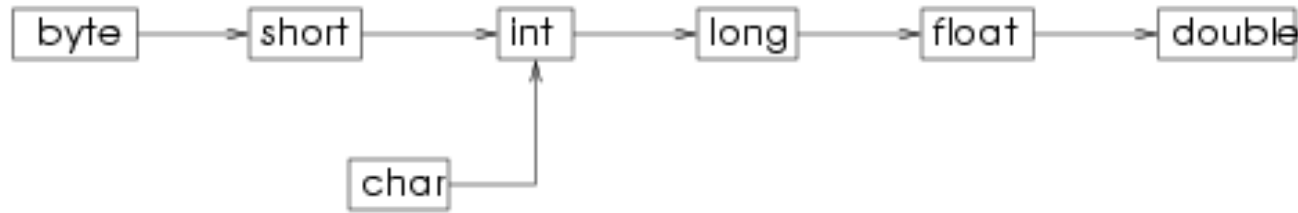
final datentyp name = wert;

Basisdatentypen

Typ	Deklaration	Zuweisung	Wertebereich	Speicher
Ganzzahlig:	short x	x = 1;	-/+ 32767	16 Bit
	int x;	x = 10000;	-/+ 2147483647	32 Bit
	long x	x = 100000;	-/+ 9223372036854775807	64 Bit
Character:	char x;	x = 'c';	'u0000'...'uffff'	16 Bit
Gebrochen- rational:	float x;	x = 5.7899;	-/+ 10 ³⁸	32 Bit
	double x;	x = 2.3;	-/+ 10 ³⁰⁸	64 Bit
Boolean:	boolean x;	x = true;	true, false	1 Bit

Typkonvertierungen (type casting)

Automatisch mit der Pfeilrichtung:



Manuell gegen die Pfeilrichtung:

⇒ (int) = Integerwert einer Zahl

```
double y = 2.5;
```

```
int x;
```

```
x = (int)y;           →           x = 2
```

⇒ (double) = Gleitkommawert einer Zahl

```
double x;
```

```
x = (double)2;        →           x = 2.0
```

Arithmetische Operationen

Addition: $+$ $\rightarrow$ $x = 2.3 + x;$ oder $x += 2.3;$

Subtraktion: - $\rightarrow$ $x = x - 2.3;$ oder $x -= 2.3;$

Multiplikation: $*$ $\rightarrow$ $x = 5 * x;$ oder $x *= 5;$

Division: double x;

`x = 5 / 2;` $\rightarrow$ `x = 2;` `// ganzzahlige Division`

`x = 5.0 / 2;` $\rightarrow$ `x = 2.5;`

[illegible]

Beispiel Programm

```
public class EinfacheRechnung {  
    public static void main(String[] args)  
    {  
        int x, y;  
        x = 10;  
        y = 15 + 13 / 5 * (7 + 6);  
        System.out.print("Das Resultat: ")  
        System.out.print(x+y);  
        System.out.println(" ");  
    }  
}
```

Vergleiche + Logische Operatoren

gleich	:	<code>==</code>	→	<code>(5 == 6)</code>	→	<code>false</code>
ungleich	:	<code>!=</code>	→	<code>(5 != 6)</code>	→	<code>true</code>
größer als oder gleich	:	<code>>=</code>	→	<code>(5 >= 6)</code>	→	<code>false</code>
kleiner als oder gleich	:	<code><=</code>	→	<code>(5 <= 6)</code>	→	<code>true</code>
und	:	<code>&&</code>	→	<code>(5 <= 6) && (5 == 8)</code>	→	<code>false</code>
oder	:	<code> </code>	→	<code>(5 <= 6) (5 == 8)</code>	→	<code>true</code>

Tipps und Tricks (2)

⇒ **Kommentare:** werden vom Compiler ignoriert

Zeilenkommentar:

// das ist ein Kommentar

Standard (mehrere Zeilen):

/ Kommentarzeile eins
zeile zwei */*

Eigene Dokumentation (javadoc): */***

** Diese Methode kann Zahlen addieren.
* javadoc wandelt das dann in HTML um.
/

Tipps und Tricks (3)

⇒ Für mehr Übersichtlichkeit Programmzeilen entsprechend einrücken.

Unübersichtlich:

```
if ((a==b) &&  
(a>0)) { y++;z=a*  
(b*3.0)/x-s1 , a+=1;  
x=Math.max(a , b);}
```

Besser:

```
if ((a==b) && (a>0))  
{  
    y++;  
    z=a*(b*3.0)/x-s1 ;  
    a+=1;  
    x=Math.max(a , b);  
}
```

Tipps und Tricks (4)

⇒ Bezeichner:

- Name von Programmelementen (Klassen, Attributen, Methoden, usw.)
- Regeln:
 - * erstes Zeichen: Buchstabe, Unterstrich oder Dollarzeichen
 - * danach: Buchstaben oder Zahlen
- Konventionen:
 - * Klasse: erster Buchstabe **groß** - MeineKlasse
 - * Attribute + Methoden: erster Buchstabe **klein** - meineVariable, meineMethode()
 - * Konstante: alles Großbuchstaben - MEINE_KONSTANTE

Kontrollstrukturen

Block

Begin

Anweisung 1

Anweisung 2

.

.

.

Anweisung n

End

Java:

{

Anweisung 1;

Anweisung 2;

.

.

.

Anweisung n;

}

Kontrollstrukturen (2)

Bedingte Anweisung

```
if Bedingung then
    Anweisung 1
    Anweisung 2
    ...
else-if Bedingung then
    Anweisung 1
    Anweisung 2
    ...
else
    Anweisung 1;
    Anweisung 2;
end-if
```

Java-Beispiel:

```
if (alter < 11) {
    person = "Kind";
}
else if ((alter >= 12) && (alter < 20)) {
    person = "Teenager";
}
else if (alter > 20) {
    person = "Erwachsen.";
}
else
    person = "???";
```

Kommandozeilenargumente

```
public static void main(String[] args)
{
    String parameter1 = args[0];
    Integer intObjekt = new Integer(parameter1);
    int number = intObjekt.intValue();
}
```

Integer = Wrapper Klasse

- beinhalten primitive Datentypen und hilfreiche Konstanten und Methoden
- z.B. MIN_VALUE, MAX_VALUE, POSITIVE_INFINITY, NEGATIVE_INFINITY

1.1 Übung

Schreiben Sie ein Programm, dem als Parameter eine Zahl übergeben (Kommandozeilenargument) wird und das als Ergebnis einen der folgenden Texte ausgibt:

- “Negativ” - wenn die Zahl kleiner als Null ist
- “Positiv” - wenn die Zahl größer als Null ist
- “Null” - wenn die Zahl gleich Null ist

Kontrollstrukturen (3)

Bewachte Anweisung

case (Ausdruck)

```
Bedingung 1: Anweisung 1.1;  
            ... Anweisung 1.n;  
Bedingung 2: Anweisung 2.1;  
            ... Anweisung 2.m;  
...  
Bedingung p: Anweisung p.1;  
            ... p.q;
```

end-case

Java-Beispiel:

```
switch (alter)  
{  
    case 9:  person = "Kind";  
            break;  
    case 14: person = "Teenager";  
            break;  
    case 20: person = "Erwachsener";  
            break;  
    default: person = "???" ;  
}
```

1.2. Übung

Schreiben Sie ein Programm, das das Spiel “FritzQuack” nachbildet. Es zählt bis 100 und ersetzt dabei jedes Vielfache von fünf durch das Wort “Fritz”, jedes Vielfache von sieben mit “Quack” und jedes Vielfache von fünf und sieben mit “FritzQuack”.

- a) mit Hilfe der `if-else`-Anweisung
- b) mit Hilfe der `switch`-Anweisung

Schleifen

Die abweisende Schleife

while (Bedingung) **do**

Anweisung 1

Anweisung 2

.

.

.

Anweisung n

end-while

Java-Beispiel:

Ausgabe der Zahlen 1 bis 10

```
int i = 0;
while (i++ < 10) {
    System.out.println(i);
}
```

Schleifen (2)

Die nicht-abweisende Schleife

do

Anweisung 1

Anweisung 2

.

.

.

Anweisung n

while (Bedingung)

Java-Beispiel:

Ausgabe der Zahlen 1 bis 10

```
int i = 0;
do {
    System.out.println(i);
}
while (i++ < 10)
```

Schleifen (3)

Die Zählschleife

```
for (Initialisierung; Bedingung; Inkrementanweisung)
{
    Anweisung 1
    Anweisung 2
    ...
    Anweisung n
}
```

Java-Beispiel: Ausgabe aller Zahlen von 1 bis 10

```
for (int i = 1; i <= 10; i++) {
    System.out.println(i);
}
```

Inkrementoperator: $i++ \rightarrow i=i+1$

Dekrementoperator: $i-- \rightarrow i=i-1$

1.3 Übung

Die Fakultät einer Zahl berechnet sich nach der folgenden Formel:

$$n! = 1 * 2 * 3 * \dots * n$$

Schreiben Sie ein Programm, dass eine natürliche Zahl als Parameter einliest und die Fakultät dieser Zahl als Ergebnis ausgibt.

1.4. Übung (Zusatz)

Um die Fakultät zu berechnen soll die folgende Formel

$$n! = n * (n - 1)!$$

in einer rekursiven Methode implementiert werden.

Aufruf von Methoden:

```
public class Klasse {  
    public static int addiere(int x, int y) {  
        return x + y;  
    }  
    public static void main(String[] args) {  
        int a = 10, b = 13;  
        System.out.println("Ergebnis: " + addiere(a, b));  
    }  
}
```