

# **EINI LW/WiMa**

**Einführung in die Informatik für  
Naturwissenschaftler und  
Ingenieure**

**Vorlesung      2 SWS      WS 14/15**

**Dr. Lars Hildebrand**

**Fakultät für Informatik – Technische Universität Dortmund**

**[lars.hildebrand@tu-dortmund.de](mailto:lars.hildebrand@tu-dortmund.de)**

**<http://ls1-www.cs.tu-dortmund.de>**

▶ **Kapitel 6**  
Objektorientierte Programmierung - Einführung

Eini LogWing /  
WiMa

**Kapitel 6**  
Objektorientierte  
Programmierung -  
Einführung

- ▶ **Unterlagen**
- ▶ Echte, Goedicke: Einführung in die objektorientierte Programmierung mit Java, dpunkt-Verlag.
  - ▶ Doberkat, Dissmann: Einführung in die objektorientierte Programmierung mit Java, Oldenbourg-Verlag, 2. Auflage.

**In diesem Kapitel:**

- **Prolog**
- Einführung

## Begriffe

- ▶ Spezifikationen, Algorithmen, formale Sprachen, Grammatik
- ▶ Programmiersprachenkonzepte
- ▶ Grundlagen der Programmierung

## ▶ Algorithmen und Datenstrukturen

- ▶ Felder
- ▶ Sortieren
- ▶ Rekursive Datenstrukturen (Baum, binärer Baum, Heap)
- ▶ Heapsort

## ▶ Objektorientierung

- ▶ Einführung
- ▶ Vererbung
- ▶ Anwendung

Eini LogWing /  
WiMa

### Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

In diesem Kapitel:

- Prolog
- Einführung

# Gliederung

---

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

- ▶ Elemente höherer Programmiersprachen
- ▶ Warum Klassen & Objekte?
- ▶ Aufbau eines Java-Programms
  - ▶ Syntaktische Struktur
  - ▶ Syntaxdiagramme
  - ▶ Attribute
  - ▶ Methoden
  - ▶ Instanziierung
- ▶ Details der objektorientierten Programmierung in Java

### In diesem Kapitel:

- Prolog
- **Einführung**

## Was haben wir bis jetzt?

- ▶ Einfache Anweisungen
- ▶ Zusammenfassung von mehreren Anweisungen zu einer Funktion (oder Unterprogramm oder Prozedur ...)
- ▶ Einfache Datentypen
  - ▶ Einzelne Werte werden selten betrachtet; daher: ganze Wertemengen
  - ▶ Grundvorrat an einfachen (primitiven) Wertemengen vorgegeben:
    - ein Abschnitt der ganzen Zahlen
    - eine Teilmenge der rationalen Zahlen
    - einfache Zeichen und Zeichenketten
    - Wahrheitswerte (wahr, falsch)

# Elemente von höheren Programmiersprachen

- ▶ Dazu werden die üblichen Operationen auf diesen Mengen zur Verfügung gestellt

→ Wertemengen + Operationen = Datentyp

- ▶ Primitive Wertemengen werden als Basis für **komplexe Strukturen** benutzt

- ▶ Vorschriften, wie einzelne einfache Daten zu komplexen Gebilden geformt werden

→ **komplexe Datenstrukturen:**

- ▶ Feld / Array,
- ▶ Struct / Record, ...

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

### In diesem Kapitel:

- Prolog
- **Einführung**

# Warum Klassen und Objekte

**Die Trennung von Verfahrensvorschrift (Algorithmus) und Datenstrukturen hat sich als hinderlich erwiesen.**

Eini LogWing /  
WiMa

Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

- ▶ Besonders wichtig für das Verständnis von Programmen ist die **Zusammenfassung** der **zulässigen Verfahren** und der **dazugehörigen Datenstrukturen**
- ▶ Damit können auf einzelne Objekte zugeschnittene Verfahren formuliert werden, die **spezifische Eigenschaften eines Objektes sicherstellen**
- ▶ Diese **Zusammenfassung** findet in einer **Klasse** statt.

In diesem Kapitel:

- Prolog
- **Einführung**

# Klassen von Objekten

In der Regel ist es mühsam, alle betrachteten Objekte einzeln zu beschreiben ...

- ▶ Man ist **nicht** an einer **einzelnen Person**, sondern an **allen Personen** interessiert.
  
- ▶ Daher
  - ▶ Zusammenfassung aller **gleichartigen Objekte** zu einer **Klasse von Objekten**
  - ▶ Üblicherweise werden dann in Programmen nur die
    - Definition von Klassen und
    - Verfahren angegeben,
  - ▶ wie einzelne Exemplare (Objekte, Instanzen) geschaffen werden können

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

In diesem Kapitel:

- Prolog
- **Einführung**



# Was macht Objektorientierung aus?

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

### Objekte:

- ▶ Vereinigung von Algorithmus und Datenstrukturen

### Verallgemeinerung und Abstraktion:

- ▶ Klassen von Objekten

### Neue Begriffe in diesem Umfeld:

- ▶ Objekt
- ▶ Instanz
- ▶ Klasse
- ▶ Methode
- ▶ Vererbung
- ▶ Interface
- ▶ ...

### In diesem Kapitel:

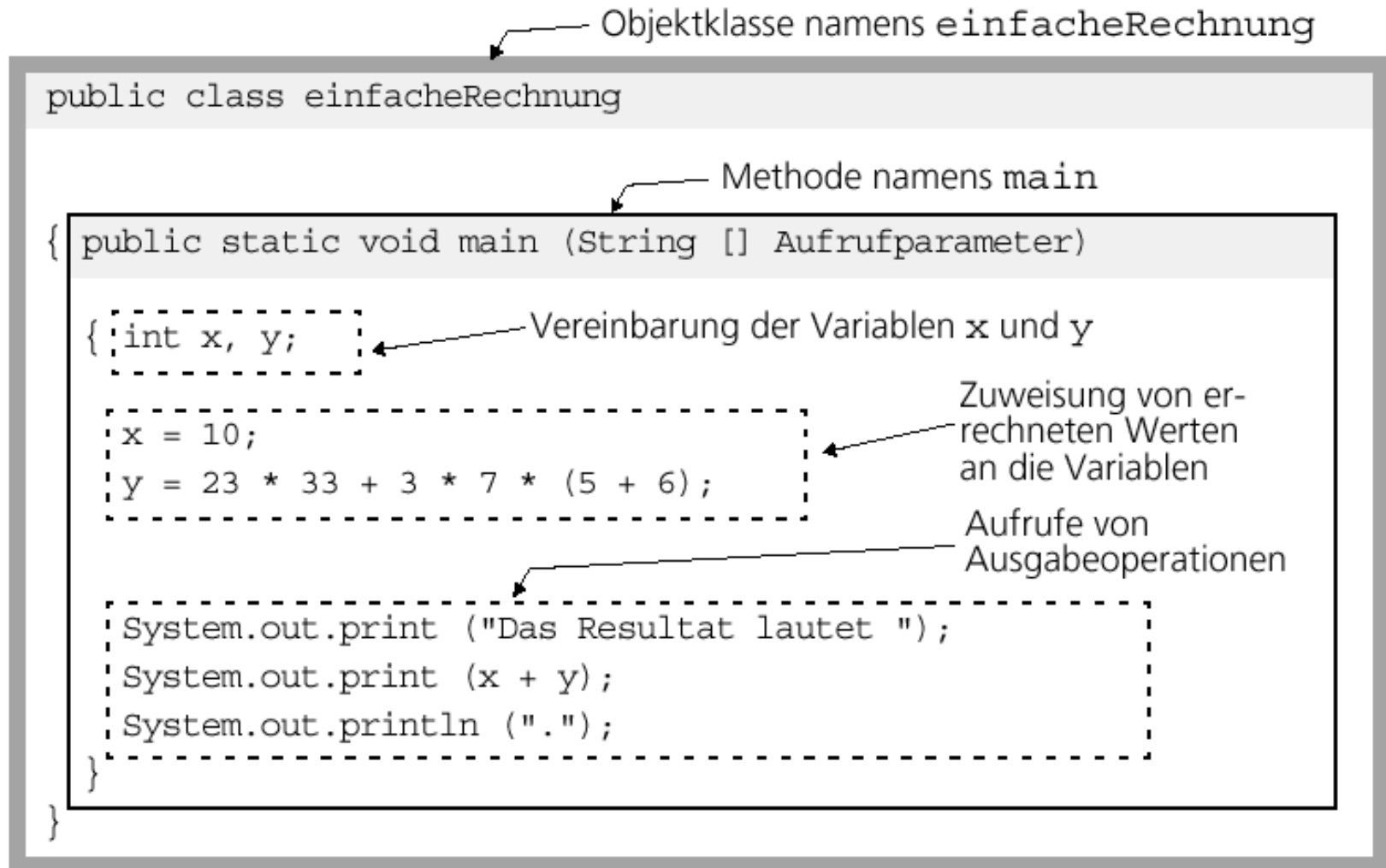
- Prolog
- **Einführung**

# Aufbau eines Java-Programms

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung



### Prog. 2-2 Erläuterung des Programms 2-1

Lehrbuch der Programmierung mit Java, Echtele Goedicke, Heidelberg, © dpunkt 2000

### In diesem Kapitel:

- Prolog
- **Einführung**

# Aufbau eines Java-Programms

## Syntaktische Struktur von (Java-) Programmen

- ▶ Die syntaktische Struktur von (Java-) Programmen ist durch Schachteln von Blöcken gegeben.
- ▶ Generell bestehen solche Schachteln aus einem **Kopf** und einem **Rumpf**:

```
public class einfacheRechnung {  
    ...  
    public static void main ( ... ) {  
        ...  
    }  
}
```

# Aufbau eines Java-Programms

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

- ▶ Java-Programm:  
**Folge von Klassendefinitionen**, die wiederum u.a. eine **Folge von Methodendefinitionen** enthalten
- ▶ Im Beispiel oben gab es:
  - ▶ Die Definition einer Klasse **einfacheRechnung**
  - ▶ Innerhalb dieser Klasse die Definition einer Methode **main**
- ▶ **main**
  - ▶ spielt eine besondere Rolle,
  - ▶ da sie die Methode ist, die automatisch als erste aufgerufen wird, wenn man das Programm startet.

### In diesem Kapitel:

- Prolog
- **Einführung**

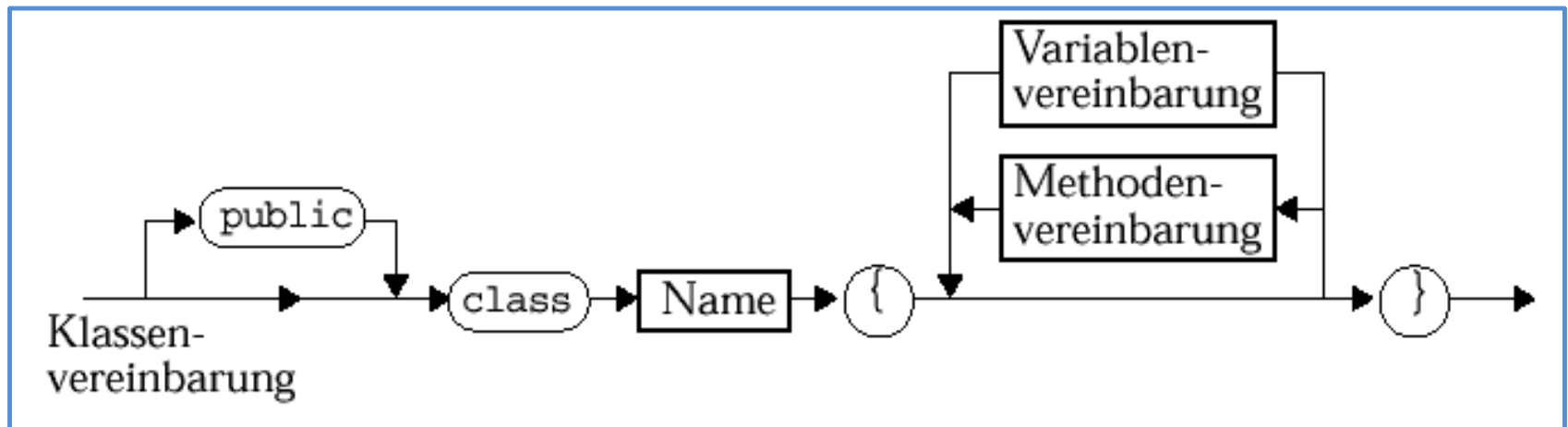
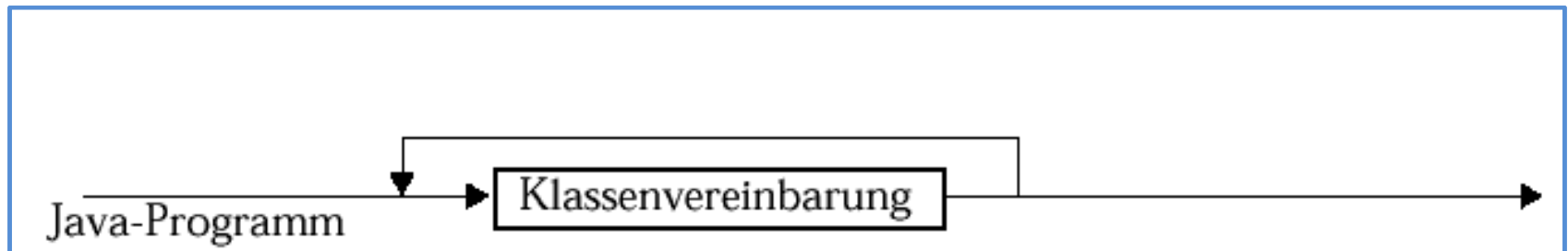
# Aufbau eines Java-Programms

## Syntax Diagramme für einfache Java-Programme (1)

Eini LogWing /  
WiMa

### Kapitel 6

Objektorientierte  
Programmierung -  
Einführung



### In diesem Kapitel:

- Prolog
- **Einführung**

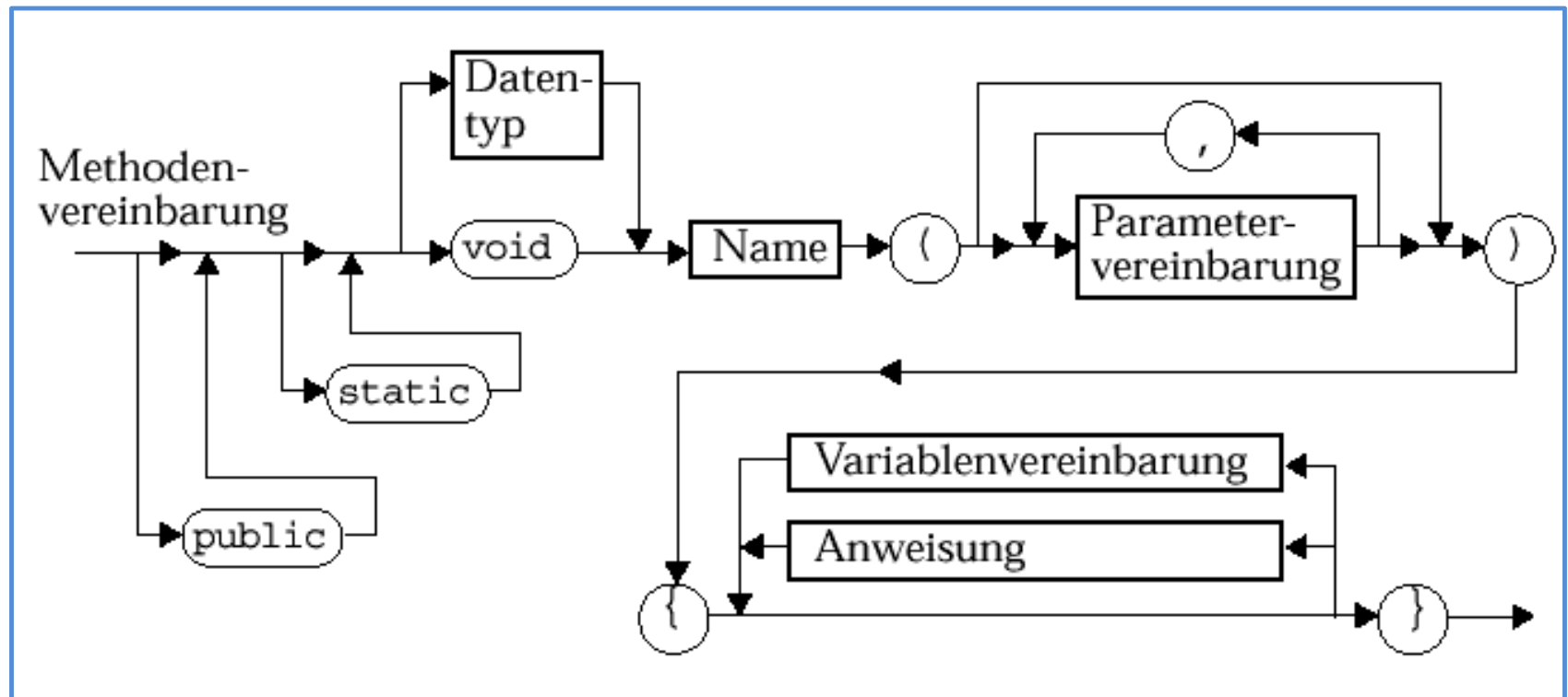
# Aufbau eines Java-Programms

## Syntax Diagramme für einfache Java-Programme (2)

Eini LogWing /  
WiMa

### Kapitel 6

Objektorientierte  
Programmierung -  
Einführung



### In diesem Kapitel:

- Prolog
- **Einführung**

# Aufbau eines Java-Programms

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

```
01 public class Girokonto {
02     private int kontostand;    // in Cent gerechnet
03
04
05     public Girokonto() {
06         kontostand = 0;
07     }
08
09     /* Einzahlen und Abheben */
10     public void zahle (int cent) {
11         kontostand = kontostand + cent;
12     }
13
14     public int holeKontostand() {
15         return (kontostand);
16     }
17 }
```

In diesem Kapitel:

- Prolog
- **Einführung**

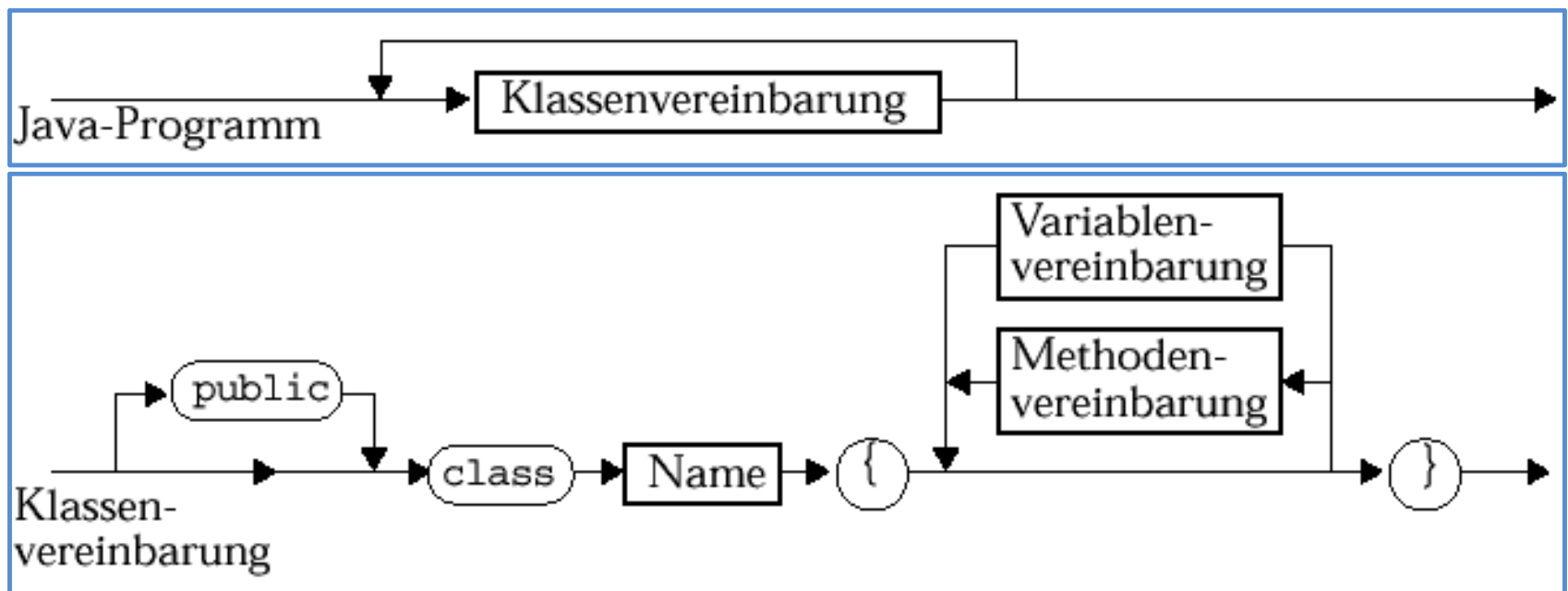
# Aufbau eines Java-Programms

```
public class Girokonto {  
    private int kontostand;  
  
    public Girokonto() {  
        kontostand = 0;  
    }  
    ...  
}
```

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung



In diesem Kapitel:

- Prolog
- **Einführung**



## Teile des Gesamtzustands eines Objekts

```
private int kontostand;  
private Girokonto meinKonto;
```

```
Modifier Datentyp attributname
```

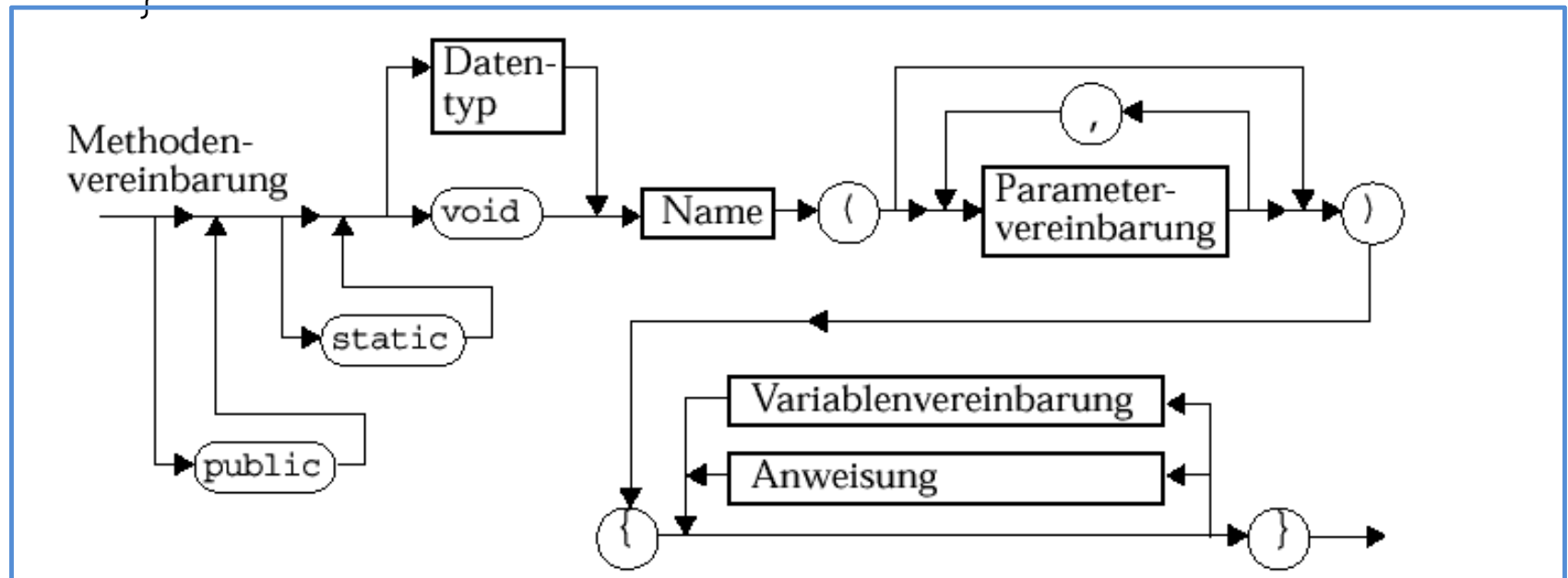
- ▶ **Verändern von Attributen**  
`kontostand = kontostand + cent;`
- ▶ **Einkapseln von Attributen** (→ Ziel: möglichst alle Attribute einkapseln)
  - ▶ **privat** d.h. nur innerhalb des Objekts sichtbar (**private**)
  - ▶ **öffentlich** d.h. auch außerhalb des Objekts sichtbar (**public**)
  - ▶ Zugriffe auf gekapselte Attribute nur durch einzelne Methoden des Objektes (getter & setter)

# Methoden (1/2) Deklaration/Definition

## Teile des Verhaltens eines Objekts

```
public void zahle (int cent) {  
    kontostand = kontostand + cent;  
}
```

```
public int holeKontostand() {  
    return (kontostand);  
}
```



Eini LogWing /  
WiMa

### Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

In diesem Kapitel:

- Prolog
- **Einführung**

# Methoden (2/2) Aufruf

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

- ▶ Aufruf von Methoden eines Objektes
  - ▶ Punkt-Notation, qualifizierter Zugriff
  - ▶ `int betrag = meinKonto.holeKontostand();`
  - ▶ `meinKonto.zahle(100);`
  
- ▶ Einkapseln von Methoden
  - ▶ privat, d.h. nur als interne Hilfs-Methode zugreifbar  
**private**
  - ▶ öffentlich, d.h. extern sichtbar  
**public**

### In diesem Kapitel:

- Prolog
- **Einführung**

# Spezielle Methode "main"

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

In diesem Kapitel:

- Prolog
- **Einführung**

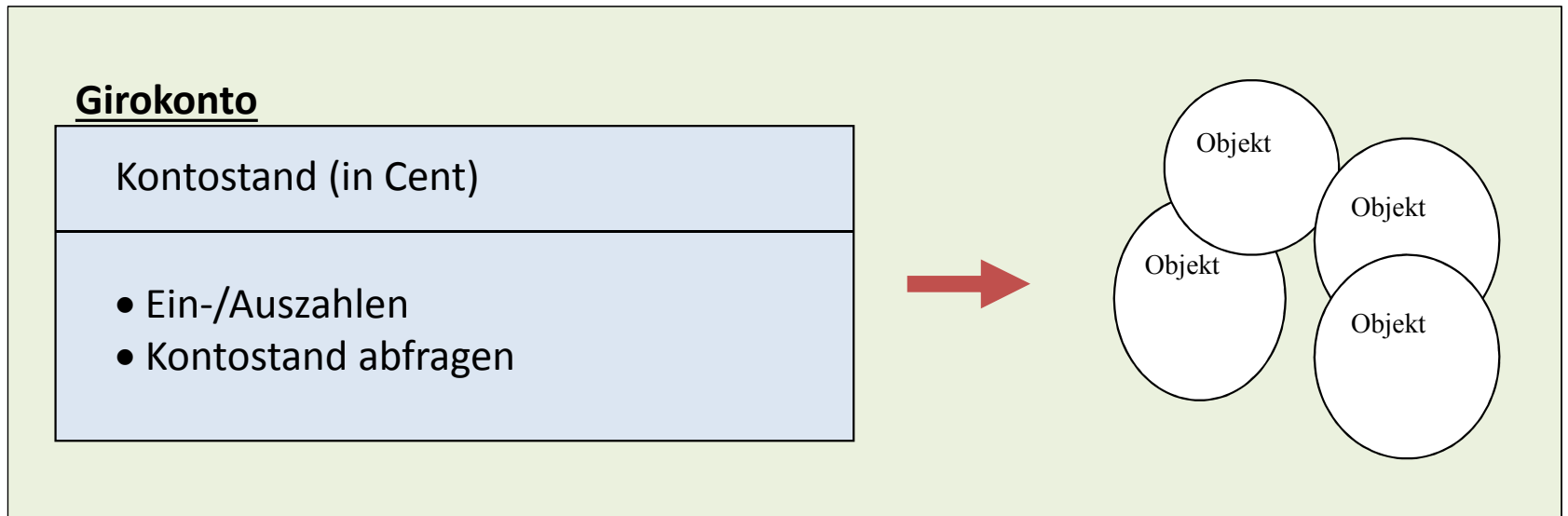
- ▶ **Einsprungspunkt** bei "Ausführen" einer Applikation
  - ▶ Übergeben von Argumenten aus der Kommandozeile möglich
  - ▶ für Testen von Klassen-Implementierungen geeignet

```
public class TestGirokonto {  
    ...  
  
    public static void main(String[] args) {  
        ... // Hier Code zum Testen einfügen  
    }  
  
    ...  
}
```

# Instanziierung

Wir haben nun die Klasse.

Wie entstehen Objekte?



## ▶ Erzeugen

```
Girokonto einKonto= new Girokonto();
```

## ▶ Löschen

- ▶ in Java: wenn Objekt nirgendwo mehr benutzt  
→ **automatische** Garbage-Collection

Eini LogWing /  
WiMa

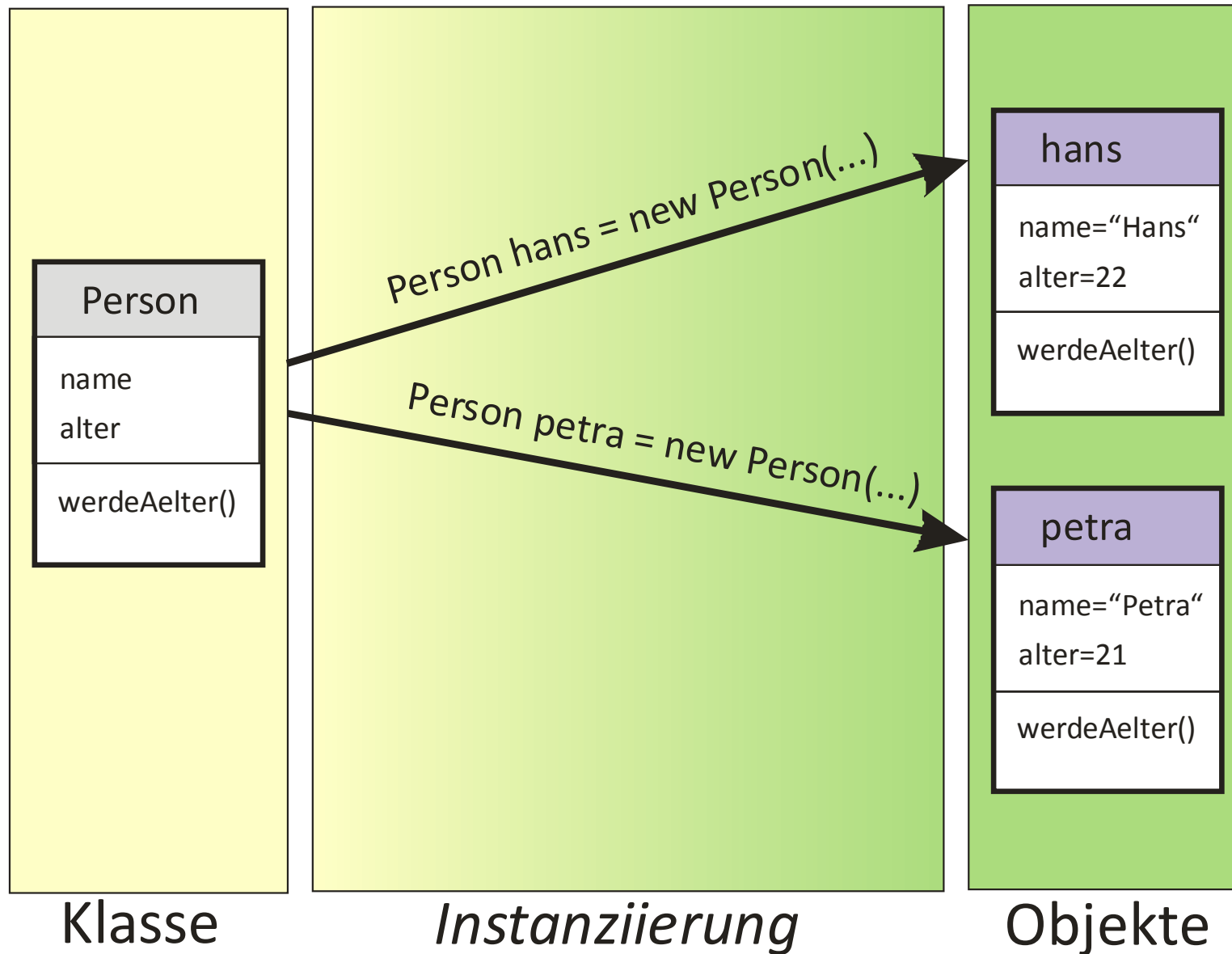
## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

In diesem Kapitel:

- Prolog
- **Einführung**

# Alternatives Beispiel



# Beispiel - Klasse Girokonto

```
public class Girokonto {
    private int kontostand;    // in Cent gerechnet

    public Girokonto() {
        kontostand = 0;
    }

    /* Einzahlen und Abheben */
    public void zahle (int cent) {
        kontostand = kontostand + cent;
    }

    public int holeKontostand() {
        return (kontostand);
    }
}
```

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

In diesem Kapitel:

- Prolog
- **Einführung**

# Beispiel - Wie können wir die Klasse testen?

▶ Testen der Klasse "Girokonto" (siehe: Methode "**main**" in der Klasse "**TestGirokonto**" )

▶ Erzeugen eines Girokonto-Objekts

▶ Nacheinander

- Einzahlen von 100 Cent,
- Abheben von 20 Cent und
- Einzahlen von 30 Cent

▶ Anschließend Kontostand ausgeben

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

### In diesem Kapitel:

- Prolog
- **Einführung**



# Beispiel - Testklasse

```
01 public class TestGirokonto {
02
03     public static void main(String[] args) {
04
05         Girokonto einKonto = new Girokonto(); // erzeuge neues
06             // Konto
07
08
09         einKonto.zahle(100); //Transaktion Einzahlung
10         einKonto.zahle(-20); //Transaktion Auszahlung
11         einKonto.zahle(30); //Transaktion Einzahlung
12
13             //gib Kontostand aus
14         int aktuellerStand = einKonto.holeKontostand(); //holen
15
16         System.out.print(aktuellerStand); //anzeigen
17
18     }
19 }
```

# Erweiterung des Bspls: Klasse Girokonto

Eini LogWIng /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

In diesem Kapitel:

- Prolog
- **Einführung**

## Motivation

- ▶ Klasse erweitern um Attribute, Methoden
- ▶ Erzeugen von Objekten und Nachrichtenaustausch (Methoden) zwischen Objekten

## Aufgabe

- ▶ Ergänzung der Funktionalität der Klasse "Girokonto"
  - ▶ zum Sperren eines Kontos: boolesches Attribut `istGesperrt` mit den Methoden `void sperre()`, `void entsperre()` und `boolean istGesperrt()`
  - ▶ zum Überschreiben des aktuellen Kontostandes mit einem übergebenen Betrag die Methode `setzeKontostand()`

# Erweiterungen

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

In diesem Kapitel:

- Prolog
- **Einführung**

```
01 public class Girokonto {
02     ...
03     private boolean istGesperrrt;
04     ...
05
06     public void sperre() {
07         istGesperrrt=true;
08     }
09
10     public void entsperre() {
11         istGesperrrt=false;
12     }
13
14     public boolean istGesperrrt () {
15         return (istGesperrrt);
16     }
17     ...
18 }
```

## Zwischenstand

- ▶ **Programm** = Mehrere Objekte, die Nachrichten austauschen
- ▶ **Klassen**: Schablonen für Objekte
  - ▶ Attribute
  - ▶ Methoden
- ▶ **Objekte**
  - ▶ Erzeugen
  - ▶ (Zerstören)
- ▶ **Nachrichtenaustausch**
  - ▶ Aufruf von Methoden eines Objektes (Punkt-Notation), z.B. `einKonto.holeKontostand()`

## Was ist anders als an bisheriger, imperativer Programmierung ?

Eini LogWing /  
WiMa

### Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

- ▶ Objekte: Attribute & Methoden zusammen
  - ▶ vorher in Structs/Records mit Zeigern auf Funktionen auch möglich, aber unüblich
- ▶ Klassen
  - ▶ vorher als Module, in denen Funktionen für bestimmte Datenstrukturen gesammelt wurden,
  - ▶ auch in imperativer Programmierung möglich
- ▶ Wesentlich ist eine andere (objektorientierte) Sicht auf eine Aufgabenstellung, die durch Sprachkonstrukte unterstützt wird.

### In diesem Kapitel:

- Prolog
- **Einführung**

# Etwas genauer: Die Details

---

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

- ▶ Konstruktoren: Aktivitäten bei Objekterzeugung
- ▶ Referenzen als Verweise auf Objekte
- ▶ Übergabe von Parametern an Methoden
- ▶ Klassenattribute / Klassenmethoden
- ▶ Namensraum
  - ▶ Überladen von Methoden
  - ▶ Überdecken von Attributen

### In diesem Kapitel:

- Prolog
- **Einführung**

# Details - Konstruktor

---

- ▶ Methode, die **automatisch** bei Erzeugung eines Objektes **aufgerufen wird**
- ▶ Wird in der Regel benutzt, um **Attribute zu initialisieren**
- ▶ Dadurch charakterisiert, dass der **Methodenname** mit dem **Klassennamen übereinstimmt**
- ▶ Konstruktoren besitzen **keinen Rückgabewert**
  - ▶ auch nicht `void`

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

### In diesem Kapitel:

- Prolog
- **Einführung**

# Details - Konstruktor: Beispiel

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

```
class Girokonto {  
    ...  
    public Girokonto() {  
        kontostand = 0;  
        ...  
    }  
}
```

gleicher  
Bezeichner

kein Rückgabotyp

Initialisierung von  
Attributen

## In diesem Kapitel:

- Prolog
- Einführung



# Details - Konstruktor

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

In diesem Kapitel:

- Prolog
- **Einführung**

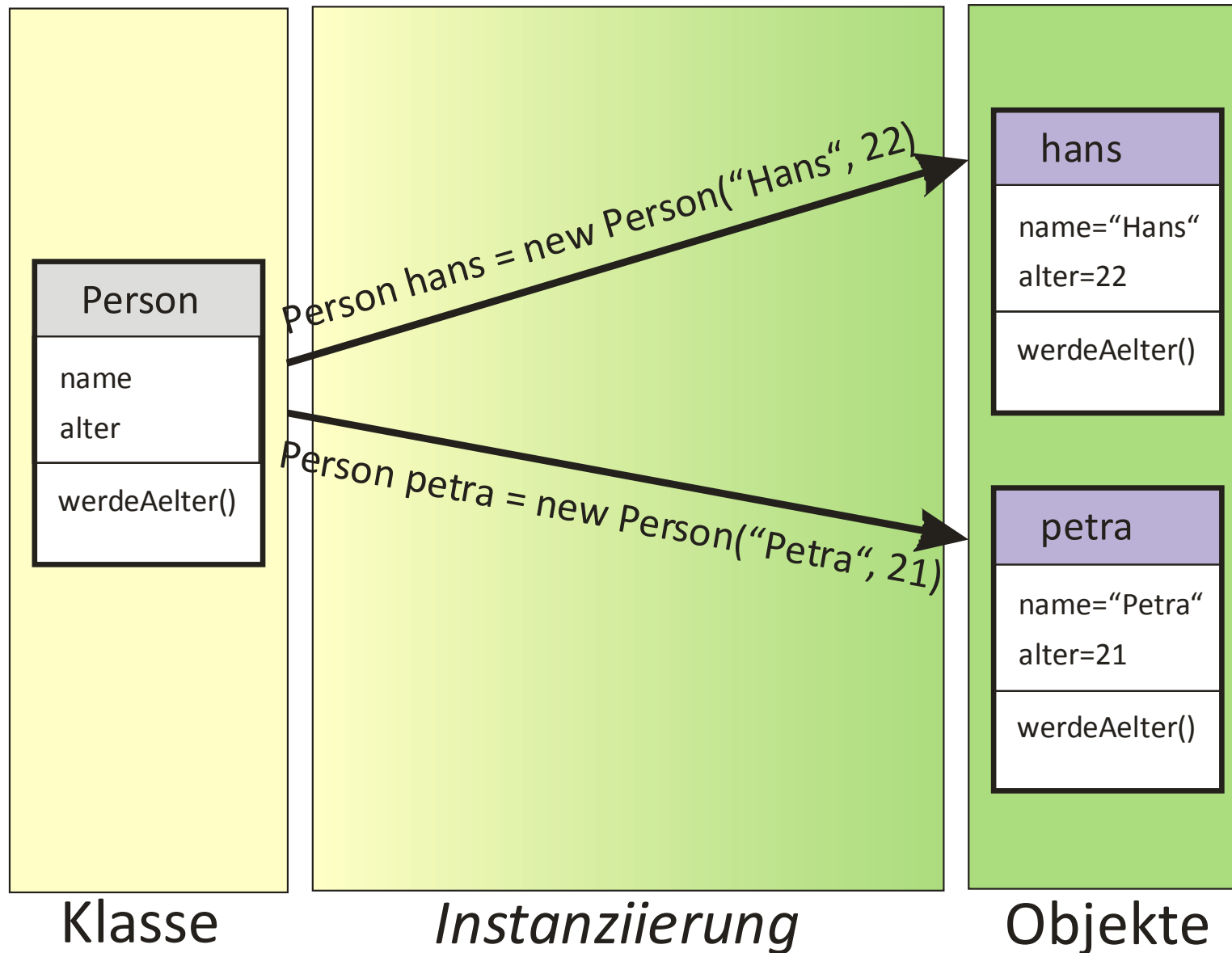
- ▶ Argumente der Konstruktormethode werden bei Objekterzeugung durch

```
new Klassenname (Argument1, Argument2,  
...)
```

an den Konstruktor übergeben.

- ▶ Anweisungen im Konstruktor werden auf neu allokiertem Speicher ausgeführt.
- ▶ Ohne Angabe eines Konstruktors ist nur der leere Konstruktor ohne Argumente automatisch definiert.
- ▶ Ist ein (nicht leerer) Konstruktor definiert, muss der leere Konstruktor explizit definiert werden, um noch benutzt werden zu dürfen.

# Alternatives Beispiel



# Details – Referenzen auf Objekte

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

- ▶ Bei primitiven Datentypen enthält eine Variable direkt den Inhalt (z. B. einen Int-Zahlenwert)
- ▶ Bei Objekten von Klassen symbolisiert die Variable nur eine Referenz (einen Verweis) auf das Objekt
- ▶ Es können auch mehrere Referenzen auf ein Objekt zeigen
- ▶ Eine Referenz kann auch leer sein, **null** ist das Schlüsselwort für die leere Referenz
  - ▶ Beispiel: `Girokonto k = null;`

### In diesem Kapitel:

- Prolog
- **Einführung**

# Details – Referenzen auf Objekte

Mit dem `==` Operator kann man zwei Referenzen auf **Gleichheit** testen

## ▶ Beispiel

```
Girokonto k1 = new Girokonto();  
  
...  
  
Girokonto k2 = k1;  
  
...  
  
if (k1 == k2)  
    System.out.println("k1, k2 verweisen auf  
                        das selbe Objekt");
```

# Details – Referenzen auf Objekte

Mit dem `==` Operator kann man abfragen, ob eine Referenz leer ist

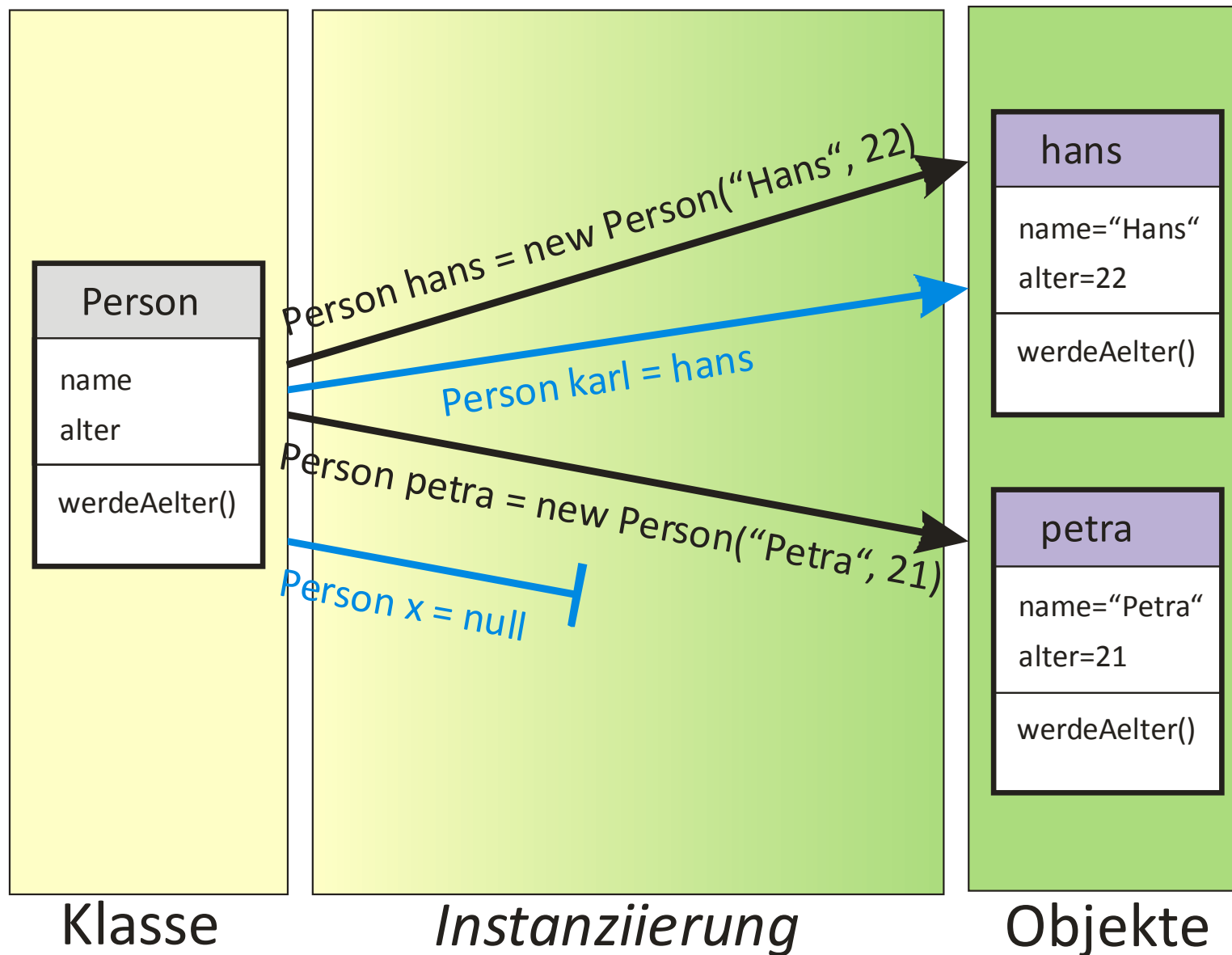
## ▶ Beispiel

```
Girokonto k=null;

...

if (k == null)
    System.out.println("k ist leere Referenz");
```

# Alternatives Beispiel



# Details - Klassenattribute und -methoden

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

### In diesem Kapitel:

- Prolog
- **Einführung**

- ▶ **Attribute** und **Methoden** gehören zu Objekten (also Instanzen) einer Klasse
- ▶ Attribute, die für jedes Objekt neue Instanzen bilden, heißen daher **Instanzvariablen** (Standardfall)
- ▶ Aber
  - ▶ Es gibt **Klassenattribute** und **-methoden**, die nicht zu Instanzen (also Objekten) gehören,
  - ▶ sondern zu den Klassen selbst

# Details – Klassenvariablen

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

- ▶ Attribute, die für jedes Objekt neue Instanzen bilden, heißen **Instanzvariablen** (Standardfall)
- ▶ Werden Attribute mit **static** gekennzeichnet, handelt es sich um **Klassenvariablen**,
  - ▶ die für die gesamte Klasse nur eine Instanz besitzen
- ▶ Klassenvariablen existieren auch ohne die Existenz eines Objektes
  - ▶ Auf sie kann durch *Klassenname.Attributname* zugegriffen werden.

### In diesem Kapitel:

- Prolog
- **Einführung**



# Details – Klassenmethoden

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

### In diesem Kapitel:

- Prolog
- **Einführung**

- ▶ Methoden,
  - ▶ die ausschließlich auf Klassenvariablen zurückgreifen
  - ▶ dürfen mit **static** gekennzeichnet werden,
  - ▶ sie heißen **Klassenmethoden**.
  - ▶ Klassenmethoden dürfen auch nur Klassenmethoden benutzen.
- ▶ Klassenmethoden **können auch ohne Existenz** eines Objektes mit *Klassenname.Methodenname(...)* aufgerufen werden.
- ▶ Die **main ()** – Methode ist eine Klassenmethode,
  - ▶ da zu Beginn noch keine Objekte erzeugt wurden.

# Details – Beispiel

```
class Demo {  
  
    int a;  
    static int b;  
  
    void test() {  
        a = 2;  
        b = 2;  
    }  
  
    static void test2() {  
        b = 3;  
    }  
}
```

Instanzvariablen  
Instanzmethoden  
Klassenvariablen  
Klassenmethoden

```
public static void main (String[] args) {  
    a = 1;           // nicht erlaubt  
    b = 1;           // erlaubt  
    test();         // nicht erlaubt  
    test2();        // erlaubt  
  
    Demo d = new Demo();  
  
    d.a = 1;        // erlaubt  
    d.b = 1;        // erlaubt  
  
    Demo.a = 1;    // nicht erlaubt  
    Demo.b = 1;    // erlaubt  
  
    Demo.test();   // nicht erlaubt  
    Demo.test2(); // erlaubt  
  
}
```

# Details – Überladen von Methoden

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

- ▶ Wunsch für Methoden/Attribute
  - ▶ Gleiche Namen für gleiche Zwecke
  - ▶ z.B. Girokonto erzeugen ohne/mit einem initialen Kontostand
- ▶ Benutzt werden sollen Methoden mit gleichen Namen
- ▶ Problem: Wie diese Methoden auseinanderhalten?
- ▶ Idee: Unterscheide anhand Parameter-Typen:
  - ▶ Methoden gleichen Namens müssen sich also im Typ von mindestens einem Parameter oder in der Anzahl der Parameter unterscheiden

```
zahle (int betrag)
```

```
zahle (int betrag, String verwendungszweck)
```

### In diesem Kapitel:

- Prolog
- **Einführung**

# Details – Überladen von Methoden

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

- ▶ Unterschiedliche Methoden müssen sich im Methodennamen oder in der Übergabeparameterliste (oder in beidem) unterscheiden
- ▶ Hat eine Klasse mehrere Methoden mit **identischem** Namen, nennt man diese Methode **überladen**
- ▶ In unterschiedlichen Klassen
  - ▶ dürfen auch Methoden mit identischem Namen und identischen Übergabeparameterlisten deklariert werden
  - ▶ → verschiedene Namensräume

### In diesem Kapitel:

- Prolog
- **Einführung**

# Details – Überdecken von Attributen

Verwenden von Variablen mit bereits benutzten Namen

► Zugreifen auf überdeckte Attribute über **this**:

```
public class Girokonto {
    private int kontostand;    // in Cent

    /* ... */

    public void setzeKontostand(int kontostand)
    {
        this.kontostand = kontostand;
    }
    /* ... */
}
```

Eini LogWing /  
WiMa

Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

In diesem Kapitel:

- Prolog
- **Einführung**

# Details – Überdecken von Attributen

Eini LogWIng /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

- ▶ Variablendeklarationen in Klassenmethoden überdecken die Attribute der Klasse
- ▶ Die Attribute sind nur **überdeckt, nicht überschrieben**
- ▶ Auf Attribute der Klasse kann dann über das Schlüsselwort **this** zugegriffen werden
- ▶ **this** ist eine Referenz auf das zur Methode gehörende Objekt

### In diesem Kapitel:

- Prolog
- **Einführung**

# Details – Beispiel

Überladen von Konstruktoren, Methoden

- ▶ Erweitern einer Klasse Girokonto um Attribute und Methoden
- ▶ Ziel:
  - ▶ **Übersicht behalten** durch sinnvolles, sparsames Vergeben von Methodennamen (Namen mehrfach in einer Klasse vergeben).

## Anforderung:

- ▶ Implementieren einer zweiten Konstruktor-Methode für die Klasse „Girokonto“
  - ▶ Sie soll es ermöglichen, dass schon bei der Erzeugung eines Girokonto-Exemplars ein Wert für den initialen Kontostand übergeben werden kann
  - ▶ Welchen Namen muss diese Methode tragen?
  - ▶ Der übergebene Parameter soll **kontostand** heißen
  - ▶ Es gibt in der Klasse "Girokonto" bereits ein Attribut mit diesem Namen.

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

In diesem Kapitel:

- Prolog
- **Einführung**

## Überladen einer Konstruktormethode

### ▶ Lösungsmuster:

- ▶ Konstruktor-Methode → Name identisch mit Klassenname
- ▶ Überdecken des Namens **kontostand** (als Attributname und Variablenname)

```
public class Girokonto {  
  
    /* ... */  
  
    public Girokonto(int kontostand) {  
        this.kontostand = kontostand;  
    }  
  
    /* ... */  
}
```





# Zwischenstand

Eini LogWing /  
WiMa

## Kapitel 6

Objektorientierte  
Programmierung -  
Einführung

- ▶ Programm = Mehrere Objekte, die Nachrichten austauschen
- ▶ Klassen: Schablonen für Objekte
  - ▶ Attribute
  - ▶ Methoden
- ▶ Objekte
  - ▶ Erzeugen
  - ▶ Zerstören
- ▶ Nachrichtenaustausch
  - ▶ Aufruf von Methoden eines Objektes (Punkt-Notation)  
z.B. `einKonto.holeKontostand()`

## In diesem Kapitel:

- Prolog
- **Einführung**



- ▶ Einzelheiten zu:
  - ▶ Konstruktoren
  - ▶ Referenzen,
  - ▶ Garbage Collection,
  - ▶ Übergabe von Parametern an Methoden,
  - ▶ Klassenattribute / Klassenmethoden
  
- ▶ Namensraum
  - ▶ Überladen von Methoden
  - ▶ Überdecken von Attributen

In diesem Kapitel:

- Prolog
- Einführung





## Frohe Weihnachten & guten Rutsch!

### Nächste Termine

- ▶ Nächste Vorlesung – WiMa 8.1.2015, 08:15
- ▶ Nächste Vorlesung – LogWIng 9.1.2015, 08:30
- ▶ Nächsten Übungen/Praktika ab 5.1.2015

