Fachhochschule Dortmund
University of Applied Sciences and Arts

Einführung in die Programmierung
WS 2020/21

VL08 - Objekte und Klassen 1

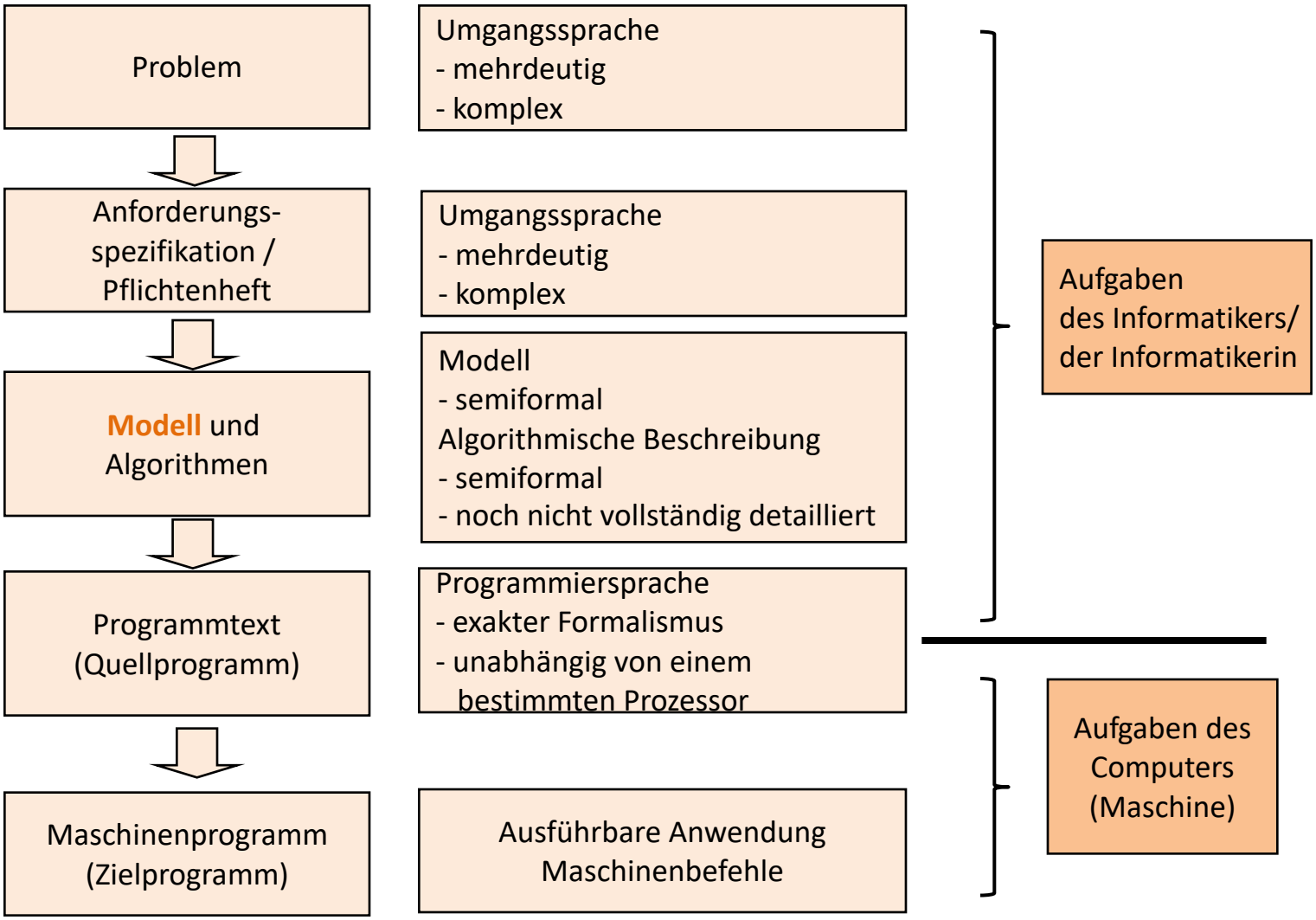
Inhalt

- Einführung
- Objekte in der Modellierung
- Klassen in der Modellierung
- Klassen in Java
- Klassen als Referenzdatentyp
- Konstruktoren in Java
- Objekte in Java

EINFÜHRUNG

Einführung

Wiederholung: Vorgehensweise bei der Programmierung



Einführung
Modellierung

Modellieren heißt, von der realen Welt zu abstrahieren und auf das zu fokussieren, was für die Aufgabe relevant ist

Beispiel: Steuerverwaltung

Reale Welt → Modell

Person1:	
Name	= Schumann
Vorname	= Clara
Steuerklasse	= 1

Person2:	
Name	= Schumann
Vorname	= Philip
Steuerklasse	= 3

Einführung

Unified Modeling Language (UML)

- UML ist eine graphische Modellierungssprache zur Erstellung objektorientierter Modelle für die Analyse und den Entwurf von objektorientierter Software
- UML wird von der Object Management Group (OMG) seit den 1990er Jahren entwickelt und standardisiert
- UML ist Quasi-Standard für eine objektorientierte Notation
- In dieser Vorlesung betrachten wir nur Klassen- und Objektdiagramme.
- aktuelle Version: UML 2.5

Objekte in der Modellierung

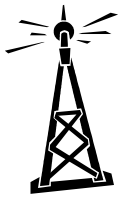
Objekte in der Modellierung

Begriff „Objekt“

Objektorientierung (Abkürzung: OO):

- Ein **Objekt** ist ein individuelles Exemplar von Dingen:
  - Gegenstände: Handy, Sendemast, Geldautomat, Kreditkarte, PKW, …
  - Personen: Professor, Student, Kunde, Mitarbeiter, Manager, …
  - Begriffe der realen Welt: Konto, Vertrag, Kinofilm, Lied, …
  - Begriffe der Vorstellungswelt: juristische Person, natürliche Person, …
- Synonyme: Exemplar, **Instanz** (engl.: instance, class instance)





- Jedes Objekt ist definiert durch seinen **Zustand**, sein **Verhalten** und seine **Identität**.

Objekte in der Modellierung

Charakteristika von Objekten – Objekt-Zustand

- Ein Objekt hat einen **Zustand** (engl. state), der durch die **Attributwerte** und **Verbindungen** zu anderen Objekten bestimmt ist.

- Beispiel: Person besitzt ein Konto

Kontonummer	= 1234567890
Kontostand	= 2500,0

Verbindung

Name	= Schumann
Vorname	= Clara
Steuerklasse	= 1

Attribute Attributwerte

Objekt-Zustand

Objekte in der Modellierung

Charakteristika von Objekten – Objekt-Verhalten

- Ein Objekt hat ein **Verhalten** (engl. behaviour), das durch eine Menge von **Methoden** (auch **Operationen** genannt) gegeben ist.

- Beispiel: Mögliche Methoden eines Kontos
  - abfragenKontonummer
  - einzahlen
  - auszahlen
  - ...

Objekte in der Modellierung

Charakteristika von Objekten – Objekt-Identität

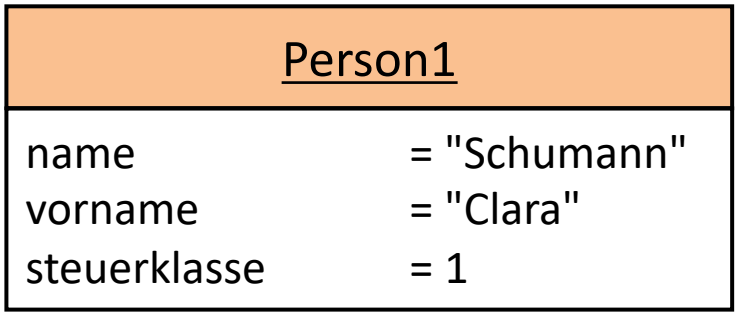
- Jedes Objekt besitzt eine **Objekt-Identität**, die es von allen anderen Objekten unterscheidet.

- Beispiel: Zwei Personen können den gleichen Namen und die gleiche Steuerklasse haben, sie sind jedoch zwei verschiedene Objekte.

Objekte in der Modellierung

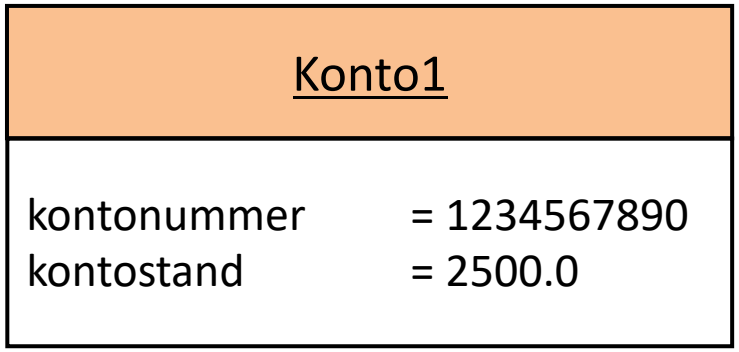
UML-Notation für Objekte

Beispiele:



Bezeichnung des Objekts (unterstrichen)

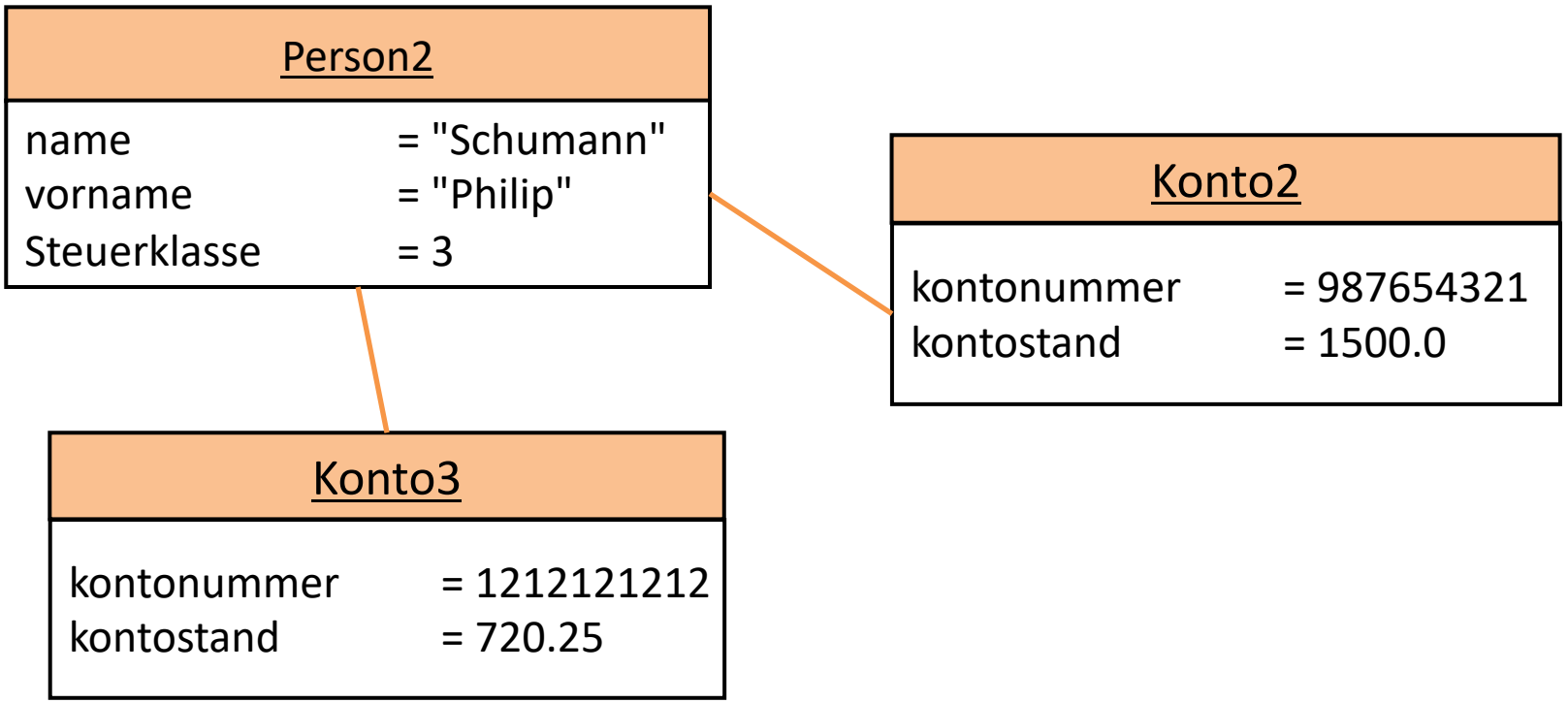
Attributliste:
Attribut = Attributwert



Objekte in der Modellierung

UML-Objektdiagramm

- Im UML-Objektdiagramm werden Objekte und ihre Beziehungen durch **Verbindungen** dargestellt
- Beispiel: Person2 besitzt Verbindungen zu zwei Konten



Klassen in der Modellierung

Klassen in der Modellierung

Begriff „Klasse“ – Charakteristika von Klassen

Objektorientierung:

- Eine **Klasse** ist eine Spezifizierung der Gemeinsamkeiten einer Menge von Objekten
  - mit denselben Attributen
  - demselben Verhalten (Methoden)
  - denselben Arten von Beziehungen (Verbindungen)
- Klassen sind also Schablonen, mit denen Objekte gleichen Typs beschrieben werden können.

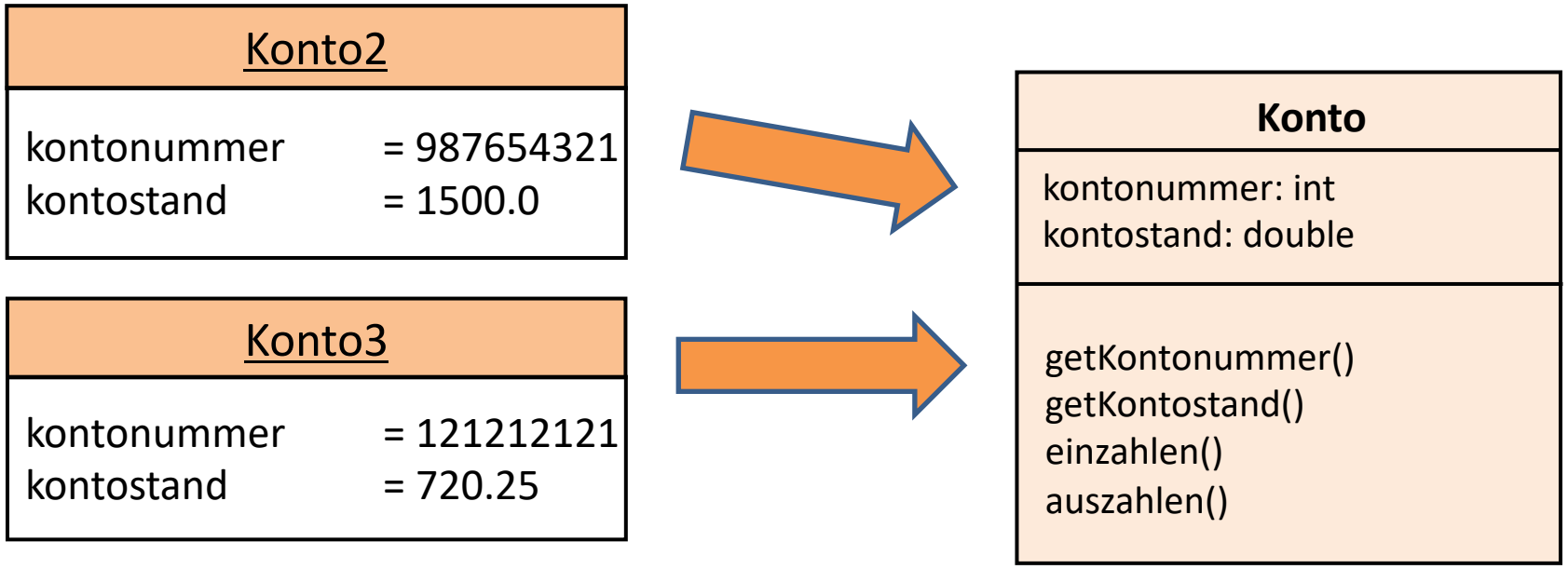
Person

Konto

Klassen in der Modellierung

Klassen als Vorlagen / Schablonen für Objekte

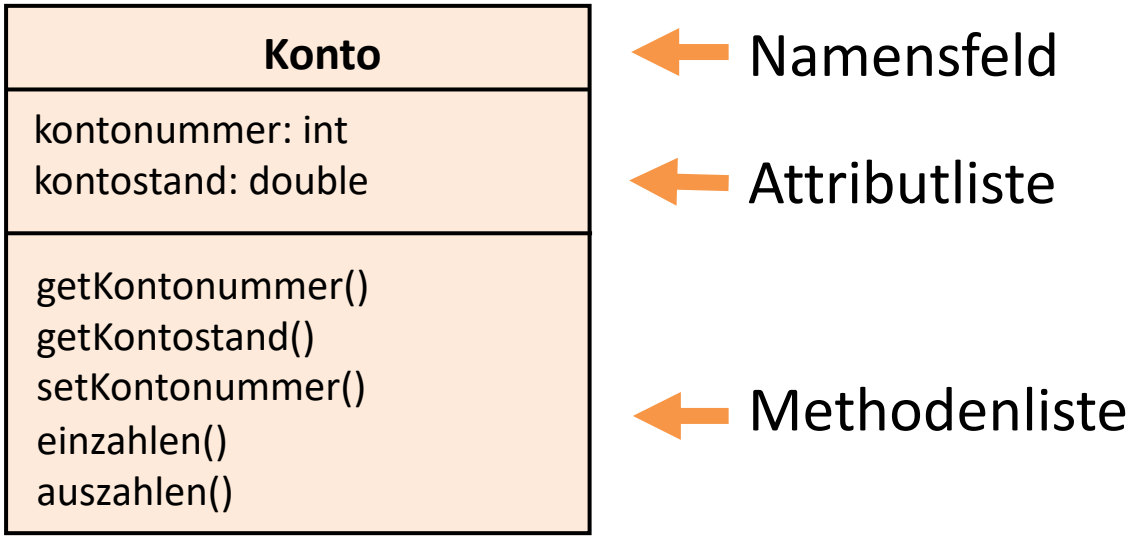
Objekte gleichen Typs besitzen identische Attribute mit (im Allgemeinen) unterschiedlichen Attributwerten und identische Methoden



Klassen in der Modellierung

UML-Notation für Klassen

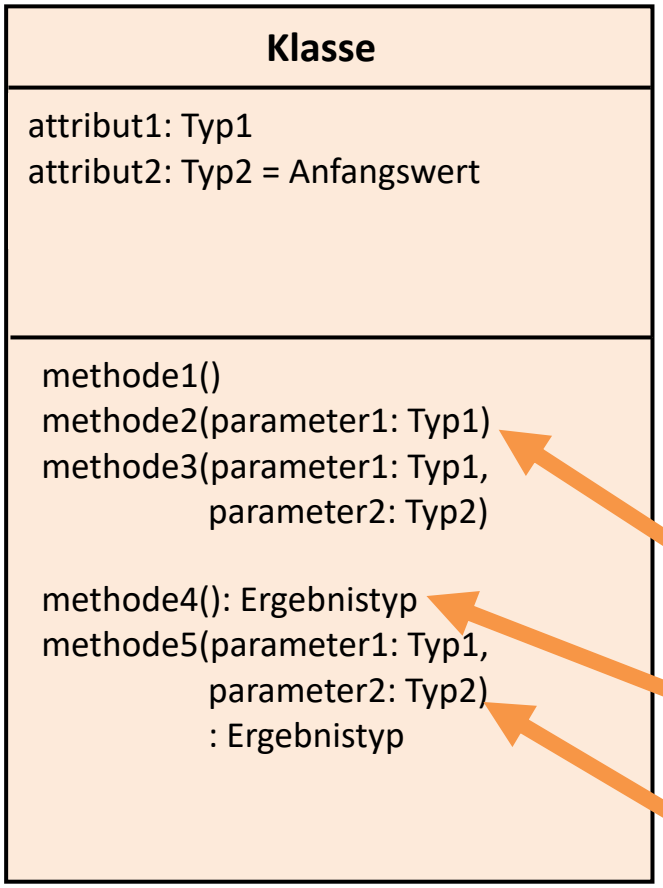
Beispiel: Klasse Konto



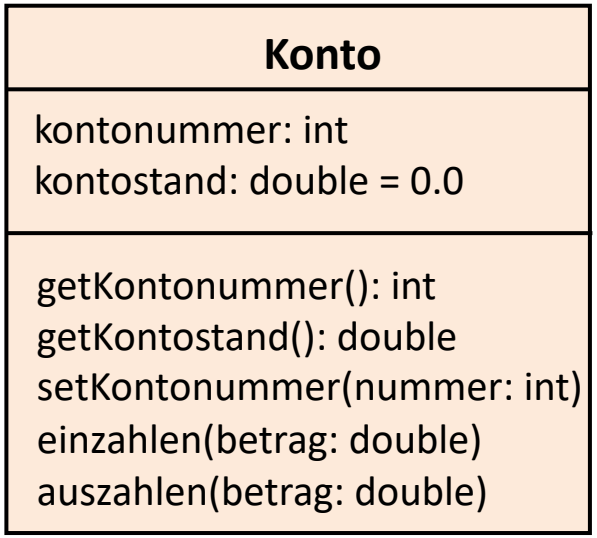
Hinweis: Methoden zum Abfragen eines Attributwertes werden üblicherweise **get**Attributname genannt

Klassen in der Modellierung
UML-Notation für Klassen

- Ausführliche Darstellung:



- Beispiel:



Notation für Prozeduren:
kein :Ergebnistyp, auch **kein :void** !

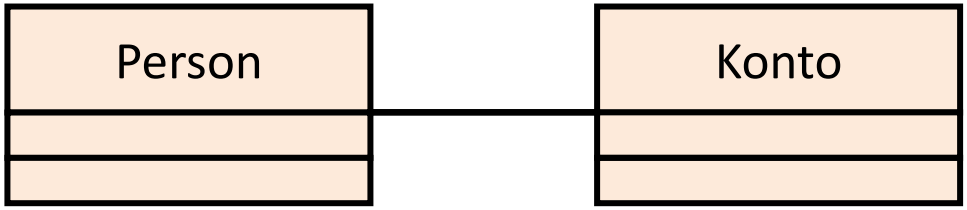
Notation für Funktionen:
:Ergebnistyp

Notation für Parameter:
Parametername: Parametertyp

Klassen in der Modellierung

UML-Klassendiagramm

- In einem **UML-Klassendiagramm** werden Klassen und ihre Beziehungen zueinander dargestellt.
- Beispiel: Personen und Konten können miteinander verbunden sein.



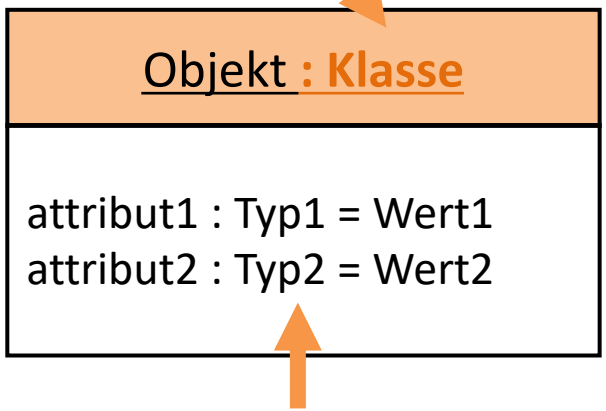
Genaueres zu Beziehungen → VL11 Assoziationen

Klassen in der Modellierung

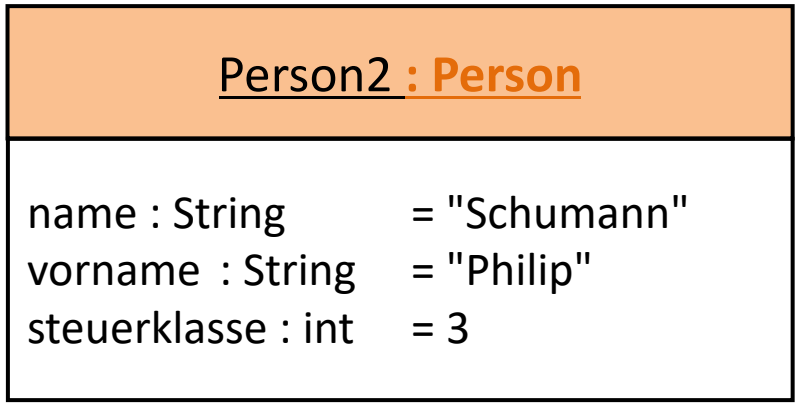
Klassen in der UML-Notation für Objekte

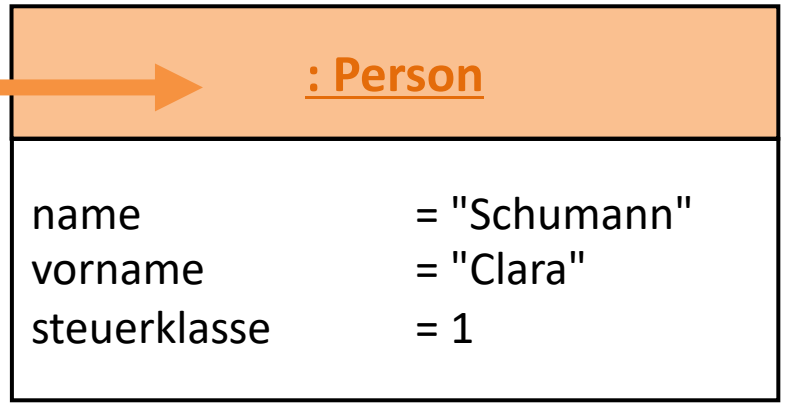
Ausführliche Darstellung:

Zusätzliche Angabe der Klasse des Objekts möglich



Zusätzliche Angabe des Attributtyps möglich:
Attribut **: Attributtyp** = Attributwert





Objektbezeichnung kann auch weggelassen werden

Klassen in der Modellierung

Namensgebung

- Klassennamen sind immer Substantive im Singular (Einzahl).
- Beispiele: Person, Handy, Mitarbeiter, Konto, …
- Zusätzlich kann dem Substantiv ein Adjektiv vorangestellt werden, Beispiele: OeffentlicheVeranstaltung, ArchiviertesDokument, …
- Attributnamen sind ebenfalls Substantive, z.B. kontostand
  Konvention in EidP: beginnen mit einem Kleinbuchstaben
- Methodennamen bestehen in der Regel aus einem Verb (z.B. einzahlen), eventuell gefolgt von einem Substantiv (z.B. getKontostand)
  Konvention in EidP: beginnen mit einem Kleinbuchstaben

Klassen in Java

Klassen in Java

Deklaration von Klassen in Java - Beispiel

Konto
kontonummer: int kontostand: double = 0.0
getKontonummer(): int getKontostand(): double setKontonummer(nummer: int) einzahlen(betrag: double) auszahlen(betrag: double)

```
class Konto
{
  // Objektattribute
  int kontonummer;
  double kontostand = 0.0;

  // Objektmethoden
  int getKontonummer()
  {
    //hier Kontonummer zurückgeben
  }
  ...
  void setKontonummer(int nummer)
  {
    //hier Kontonummer setzen
  }
  ...
  void einzahlen(double betrag)
  {
    // hier Betrag verbuchen
  }
  ...
}
```

Klassenname (engl. class identifier)

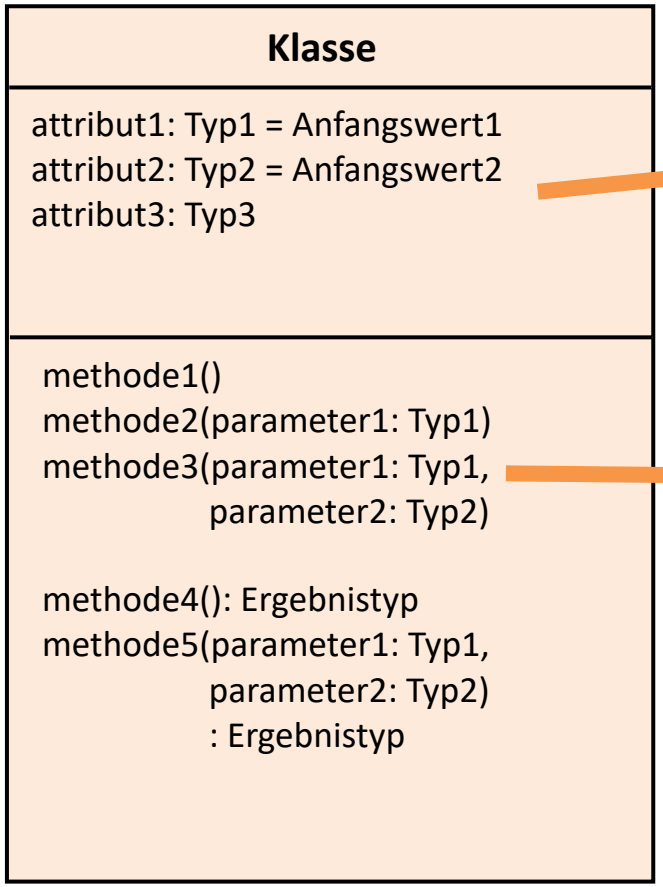
Attributdeklarationen (engl. field declarations)

Methodendeklarationen (engl. method declarations)

Klassenrumpf (engl. class body)

Klassen in Java

Deklaration von Klassen in Java – Allgemeine Struktur



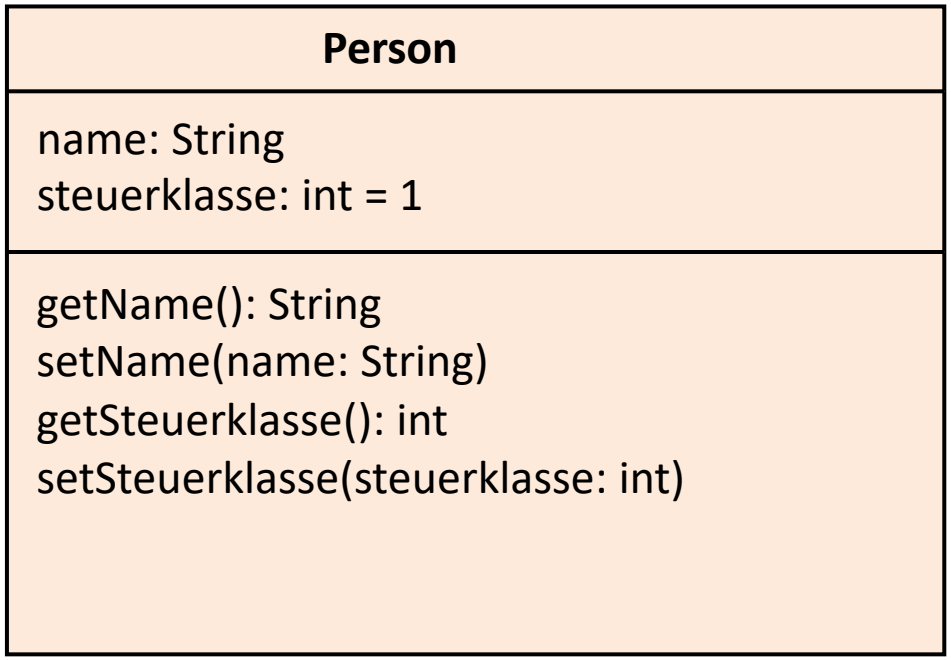

```
class Klasse
{
  // Objektattribute
   Typ1 attribut1 = Anfangswert1;
   Typ2 attribut2 = Anfangswert2;
   Typ3 attribut3;

  //  Objektmethoden
  void methode1() { // hier ergänzen }

  void methode2(Typ1 parameter1) { // hier ergänzen }

  void methode3(Typ1 parameter1, Typ2 parameter2)
  { // hier ergänzen }

  Ergebnistyp methode4() { // hier ergänzen }

  Ergebnistyp methode5(Typ1 parameter1,
                       Typ2 parameter2)
  { // hier ergänzen }
} // Ende der Klasse
```

Achtung:
Bei Objektmethoden steht in Java kein **`static`** vor dem Ergebnistyp!

Klassen in Java

Aufgabe

Übertragen Sie die UML-Beschreibung der Klasse Person in eine Java-Klasse



Klassen und Objekte in Java

Implementierung der Methoden

Beispiel:

Konto
kontonummer: int kontostand: double = 0.0
getKontonummer(): int getKontostand(): double setKontonummer(nummer: int) einzahlen(betrag: double) auszahlen(betrag: double)

Objektmethoden (Deklaration ohne **`static`**) können auf Objektattribute zugreifen!

```
class Konto
{
    // Objektattribute
    int kontonummer;
    double kontostand = 0.0;

    // Objektmethoden
    int getKontonummer() {
        return kontonummer;
    }
    double getKontostand() {
        return kontostand;
    }
    void setKontonummer(int nummer) {
        kontonummer = nummer;
    }
    void einzahlen(double betrag) {
        kontostand += betrag;
    }
    void auszahlen(double betrag) {
        kontostand -= betrag;
    }
}
```

Klassen in Java

Unterscheidung von Objektattribut und lokalen Variablen

Problem:

Namenskonflikt zwischen Objektattribut und formalem Parameter bzw. lokaler Variable

Lösung: Notation `this.`

Konto
kontonummer: int kontostand: double = 0.0
getKontonummer(): int getKontostand(): double setKontonummer(**kontonummer**: int) einzahlen(betrag: double) auszahlen(betrag: double)

```
class Konto
{
    // Attribute
    int kontonummer;
    double kontostand = 0.0;

    // Methoden
…
    void setKontonummer(int kontonummer) {
        this.kontonummer = kontonummer;
    }

…
}
```

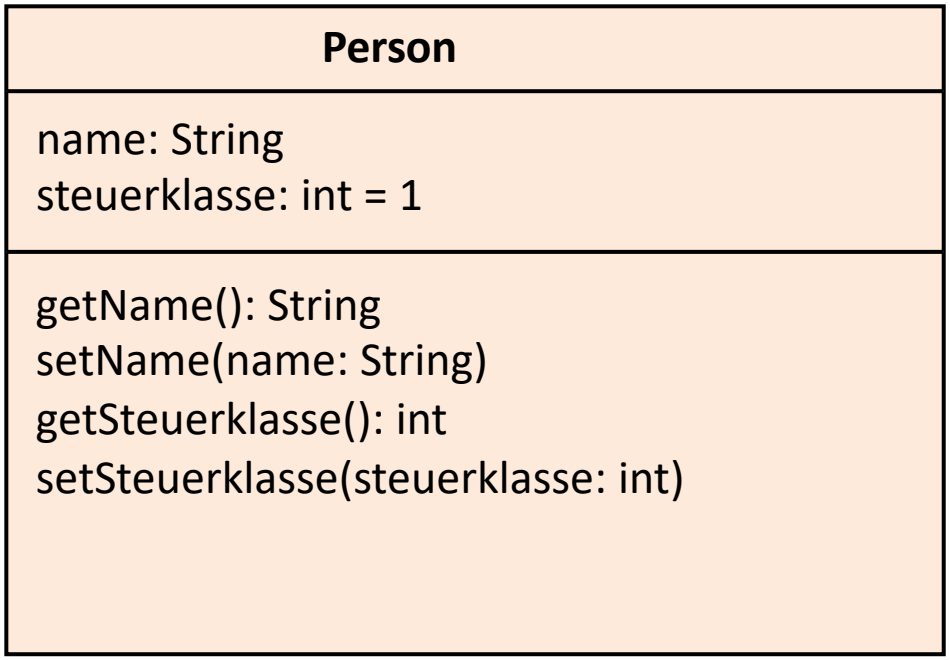
Objektattribut

Lokale Variable bzw.
Formaler Parameter

Klassen in Java

Aufgabe

Ergänzen Sie die Implementierung der Methoden in der Java-Klasse Person



Klassen in Java

Deklaration von Klassen in Java – Programmierhinweise

- Jede Klasse in einer einzelnen Datei:
  - Eine Klasse darf nicht über mehrere Dateien verteilt werden.

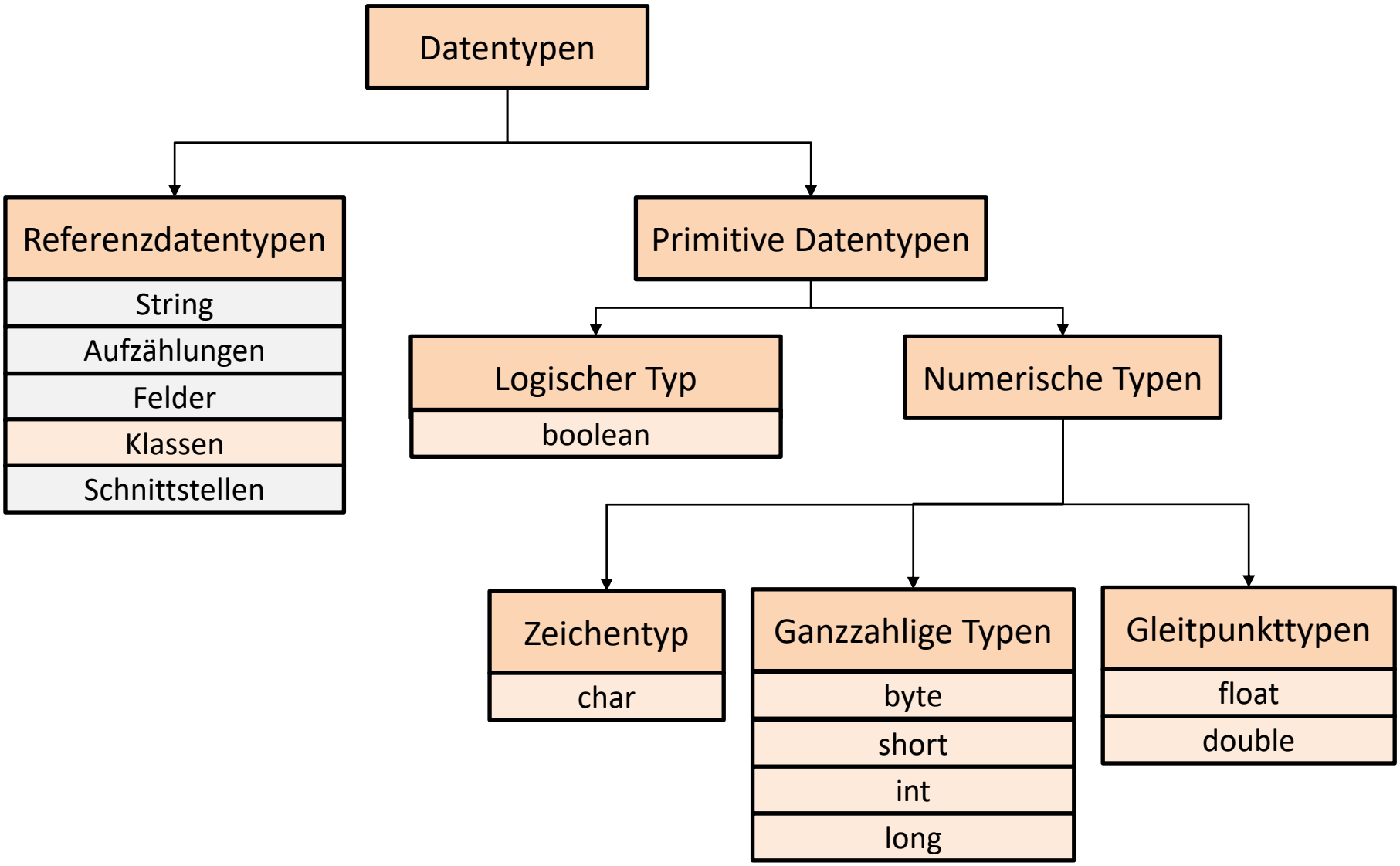
- Mehrere Klassen in einer Datei:
  - Java erlaubt unter gewissen Umständen mehrere Klassen in einer Datei (nur eine der Klassen darf als `public` gekennzeichnet sein).
  - Der Java-Compiler erzeugt von jeder Klasse, die in einer Datei abgelegt ist, eine gesonderte Datei mit der Endung `.class`.

- Empfehlung:
  - Nur eine Klasse pro Datei programmieren.
  - Der Name der Klasse und der Name der Datei sollten übereinstimmen (bei Klassen, die als `public` gekennzeichnet sind, zwingend erforderlich!).

Klassen als Referenzdatentyp

Klassen als Referenzdatentyp

Übersicht



Klassen als Referenzdatentyp

Referenzvariablen/-konstanten

- Variablen und Konstanten können einen primitiven Datentyp z.B. `int`, `boolean` oder `char` besitzen.
- Sie enthalten dann genau einen Wert dieses Datentyps.
- Variablen und Konstanten können auch auf Objekte verweisen (referenzieren). Sie heißen dann **Referenzvariable/-konstante**.
- Sie beinhalten die Adresse (Ort) des Objekts im Speicher.
- Alle Klassen können so als Datentyp verwendet werden.

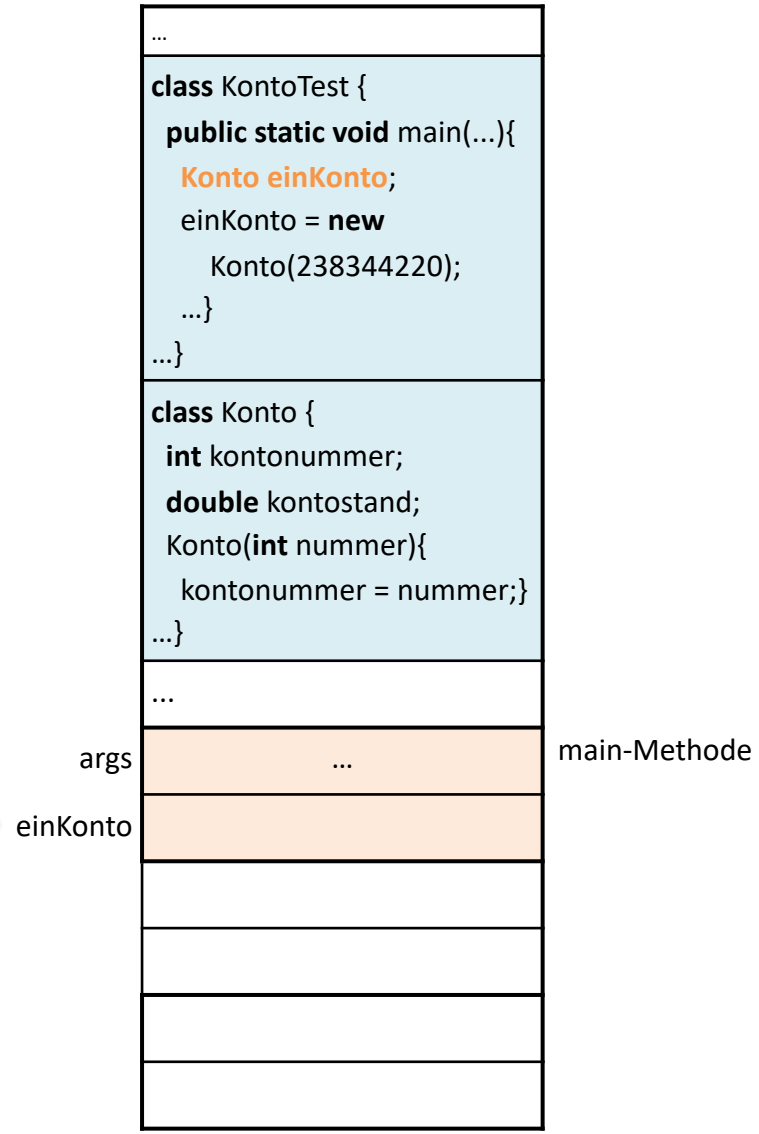
Klassen als Referenzdatentyp

Referenzvariablen

1. Schritt: Deklaration einer Referenzvariablen

- Als Typ der Variablen wird eine Klasse angegeben; eine so deklarierte Referenzvariable kann später nur auf ein Objekt von diesem Typ verweisen.
  Beispiel: `Konto einKonto;`

- Wird eine Referenvariable innerhalb einer Methode (z.B. main) deklariert, so wird bei Ausführung der Methode ein Speicherplatz für diese Variable auf dem Aufrufstapel reserviert.

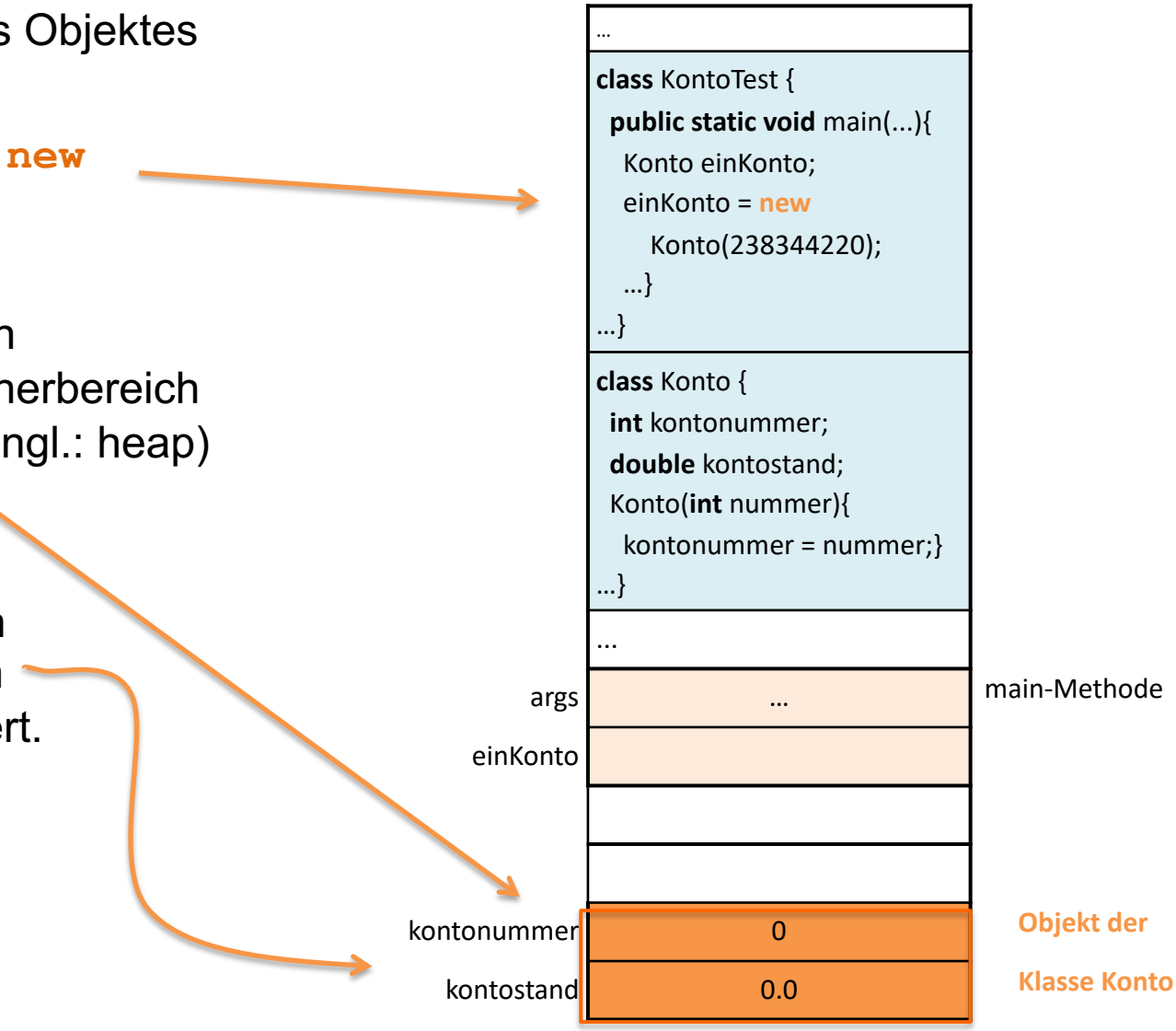


Klassen als Referenzdatentyp

Referenzvariablen

2. Schritt: Erzeugen eines Objektes

- Objekte werden durch **new** erzeugt.
- Für das Objekt wird ein entsprechender Speicherbereich im **Haldenspeicher** (engl.: heap) reserviert.
- Objektattribute werden automatisch mit einem Standardwert initialisiert.



Klassen als Referenzdatentyp

Referenzvariablen

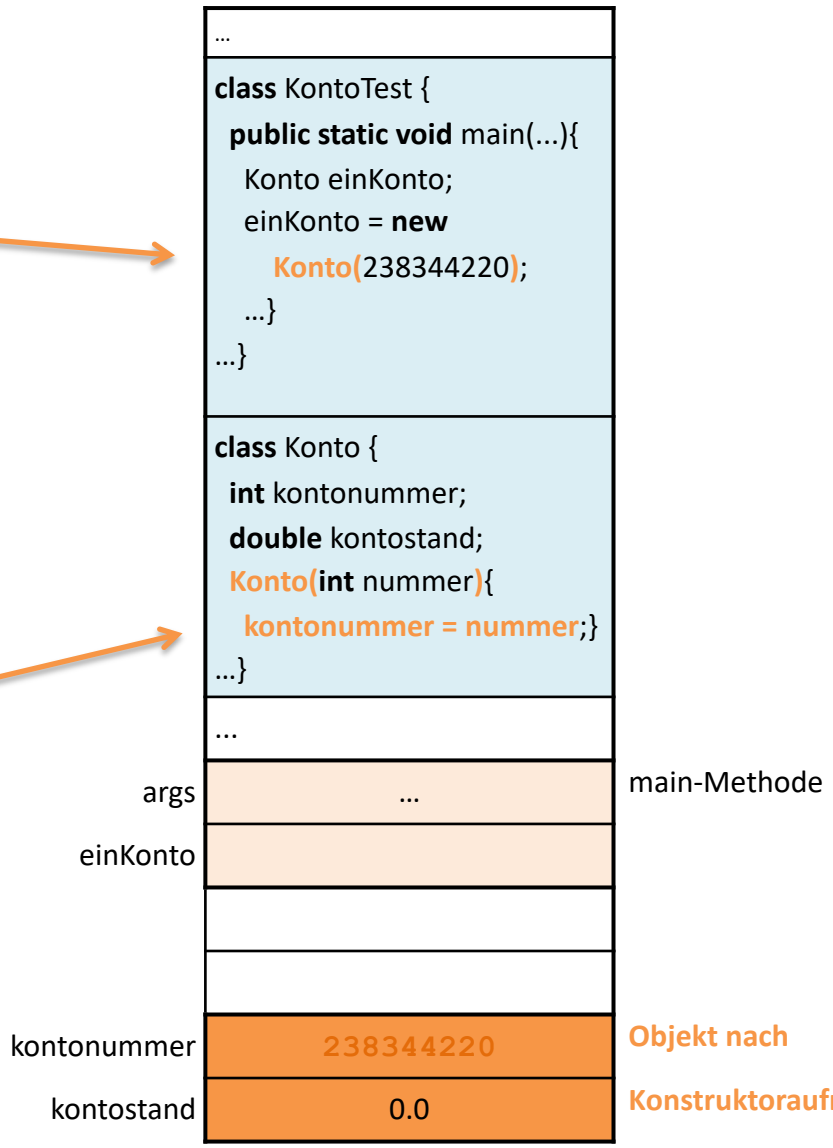
3. Schritt: Aufruf eines Konstruktors

- **Konstruktoren** sind spezielle Methoden, die unmittelbar nach dem Erzeugen eines Objektes durch **`new`** ausgeführt werden.

- Beispiel:
  `new` `Konto(238344220);`

- Im Konstruktor werden üblicherweise die Attribute erzeugter Objekte mit Werten belegt.

- Beispiel: Das Attribut `kontonummer` erhält den Wert `238344220`

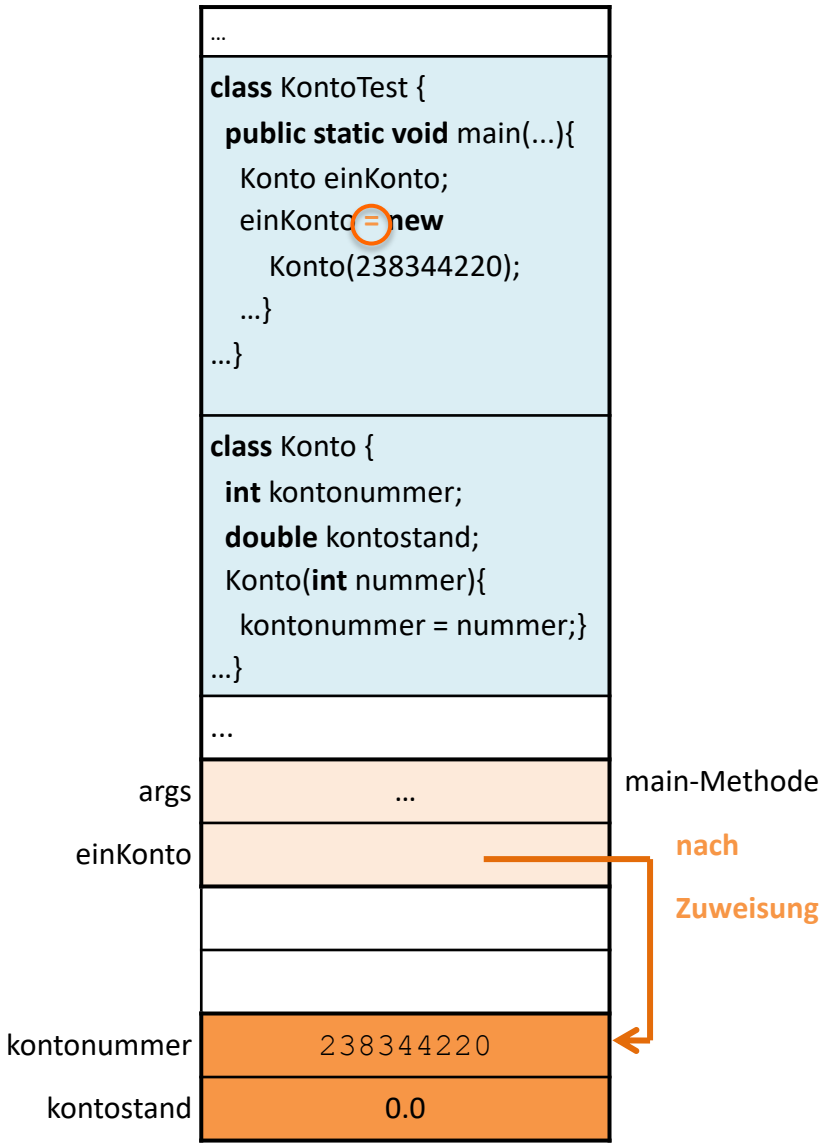


Klassen als Referenzdatentyp

Referenzvariablen

4. Schritt: Speichern der Speicheradresse

- Die Speicheradresse des erzeugten Objekts wird bei Ausführung der Zuweisung (=) in der Referenzvariablen gespeichert.
- Das Variable enthält somit eine **Referenz**, wo sich das Objekt im Speicher befindet.



Klassen als Referenzdatentyp

Speicherbereinigung

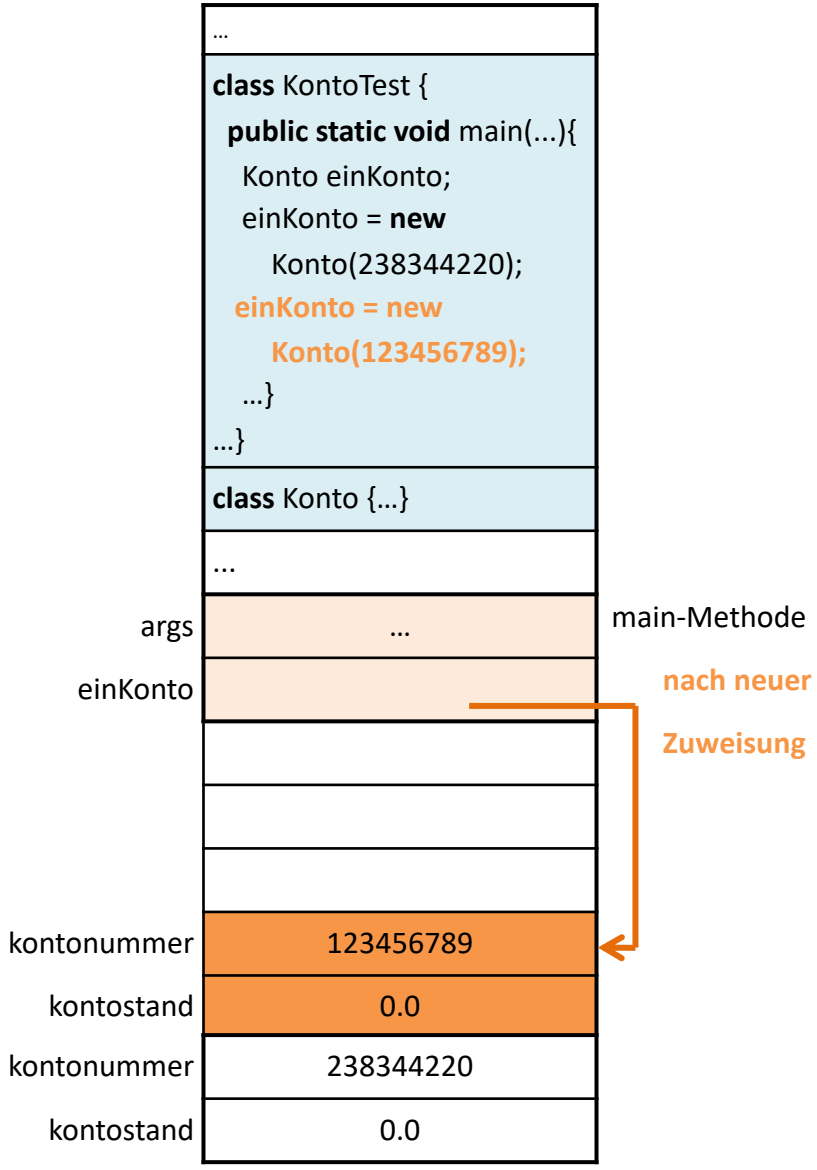
- Objekte, die im Haldenspeicher angelegt werden, müssen auch gelöscht werden.
- In Java ist dies anders als in C++ keine Aufgabe des Programmierers/der Programmiererin.
- In unregelmäßigen Abständen automatische **Speicherbereinigung** (engl.: **garbage collection**).
- Die Java-VM prüft, ob es im Haldenspeicher Objekte gibt, auf die keine Referenz mehr zeigt.
- Diese Objekte werden automatisch gelöscht.

Klassen als Referenzdatentyp

Speicherbereinigung

Beispiel:

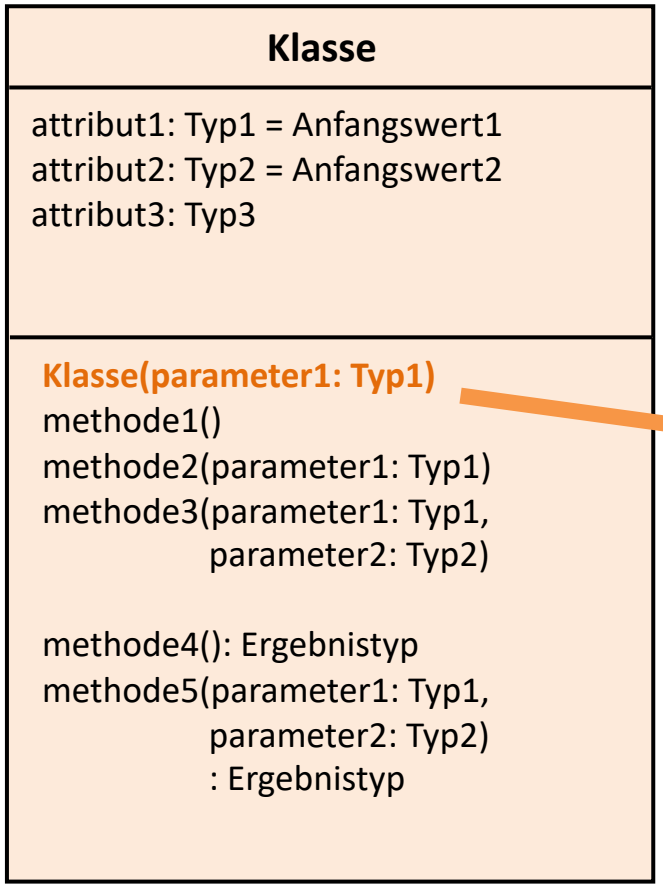
- Wenn der Referenzvariablen `einKonto` ein neuer Wert zugewiesen wird, existiert keine Referenz mehr auf das alte Objekt
- Das alte Objekt wird später von der Speicherbereinigung gelöscht



Konstruktoren in Java

Konstruktoren in Java

Deklaration von Konstruktoren




```
class Klasse
{
  // Attribute
   Typ1 attribut1 = Anfangswert1;
   Typ2 attribut2 = Anfangswert2;
   Typ3 attribut3;

  // Konstruktoren
  Klasse(Typ1 parameter1) {
     // hier ergänzen
  }

  //  Methoden
  void methode1() { // hier ergänzen }
  …

  Ergebnistyp methode5(Typ1 parameter1,
                       Typ2 parameter2)
  { // hier ergänzen }
} // Ende der Klasse
```

Konstruktoren in Java

Eigenschaften

- Der Name des Konstruktors ist identisch zum Namen der Klasse.
- Konstruktoren besitzen niemals einen Ergebnistyp.
- Konstruktoren können Parameter besitzen.
- Konstruktoren können ohne Anweisungen definiert sein.

Beispiel:

```
class Konto
{
    int kontonummer;
    double kontostand = 0.0;

    Konto(int nummer)
    {
        kontonummer = nummer;
    }
…
}
```

Konstruktoren in Java

Überladen von Konstruktoren

Beispiel:

Zwei Konstruktoren der Klasse Konto. Der erste Konstruktor besitzt einen Parameter, der zweite Konstruktor zwei Parameter.

```
class Konto
{
    int kontonummer;
    double kontostand = 0.0;

    Konto(int kontonummer)
    {
        this.kontonummer = kontoummer;
    }
    Konto(int kontonummer, double kontostand)
    {
        this.kontonummer = kontonummer;
        this.kontostand = kontostand;
    }
…
}
```

Konstruktoren in Java

Standardkonstruktor in Java

Enthält eine Klasse keine Konstruktordeklaration, erzeugt der Compiler automatisch einen Standardkonstruktor (engl.: default constructor), der keine Parameter besitzt

Objekte in Java

Objekte in Java

Zugriff auf Objektattribute in Java

- Notation: `Objektname.Attributname`

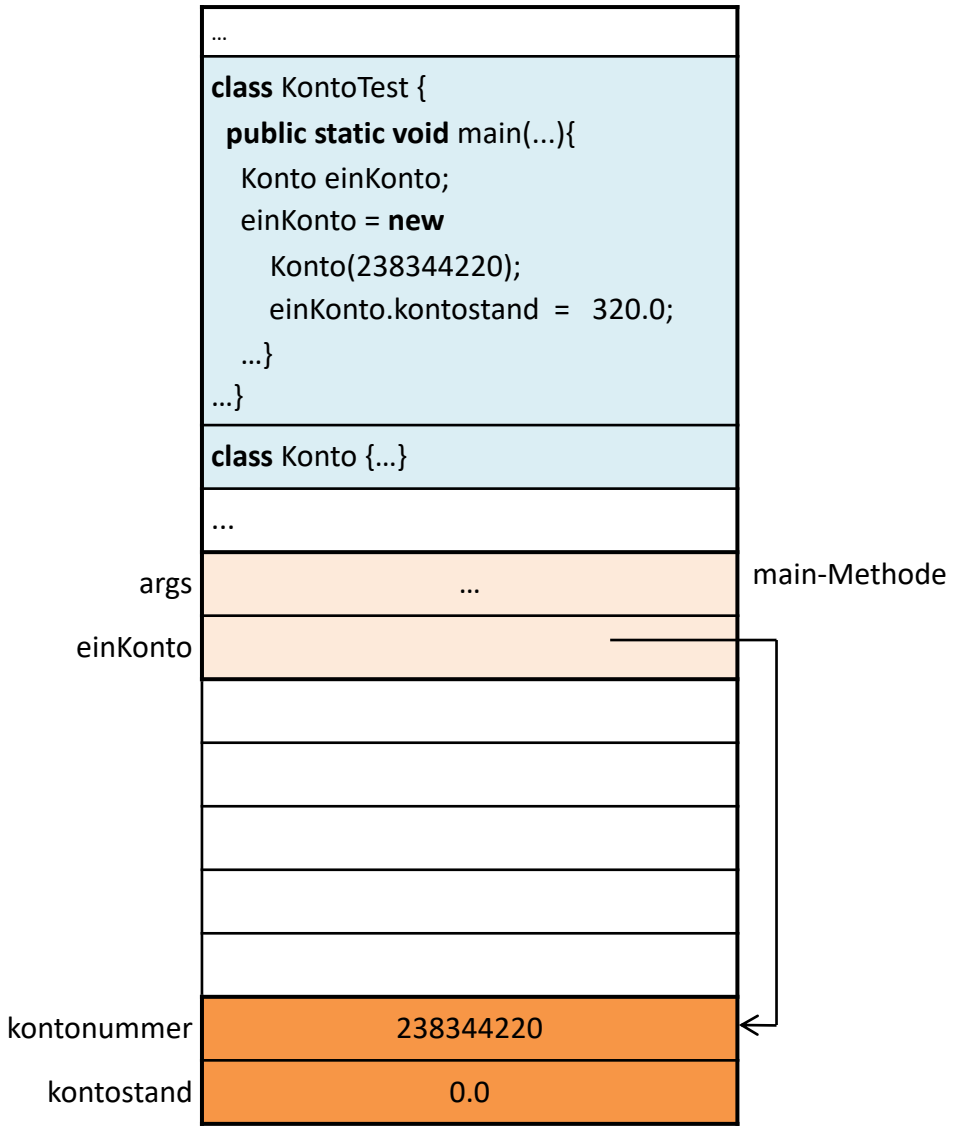
- Beispiel:

```
class KontoTest
{
    public static void main(String[] args)
    {
        Konto einKonto = new Konto(238344220);

        // Schreibender Zugriff
        einKonto.kontostand = 320.0;

        // Lesender Zugriff
        double stand = einKonto.kontostand;
        System.out.println("Kontostand = " + stand);
    }
}
```

Objekte in Java

Zugriff auf Objektattribute

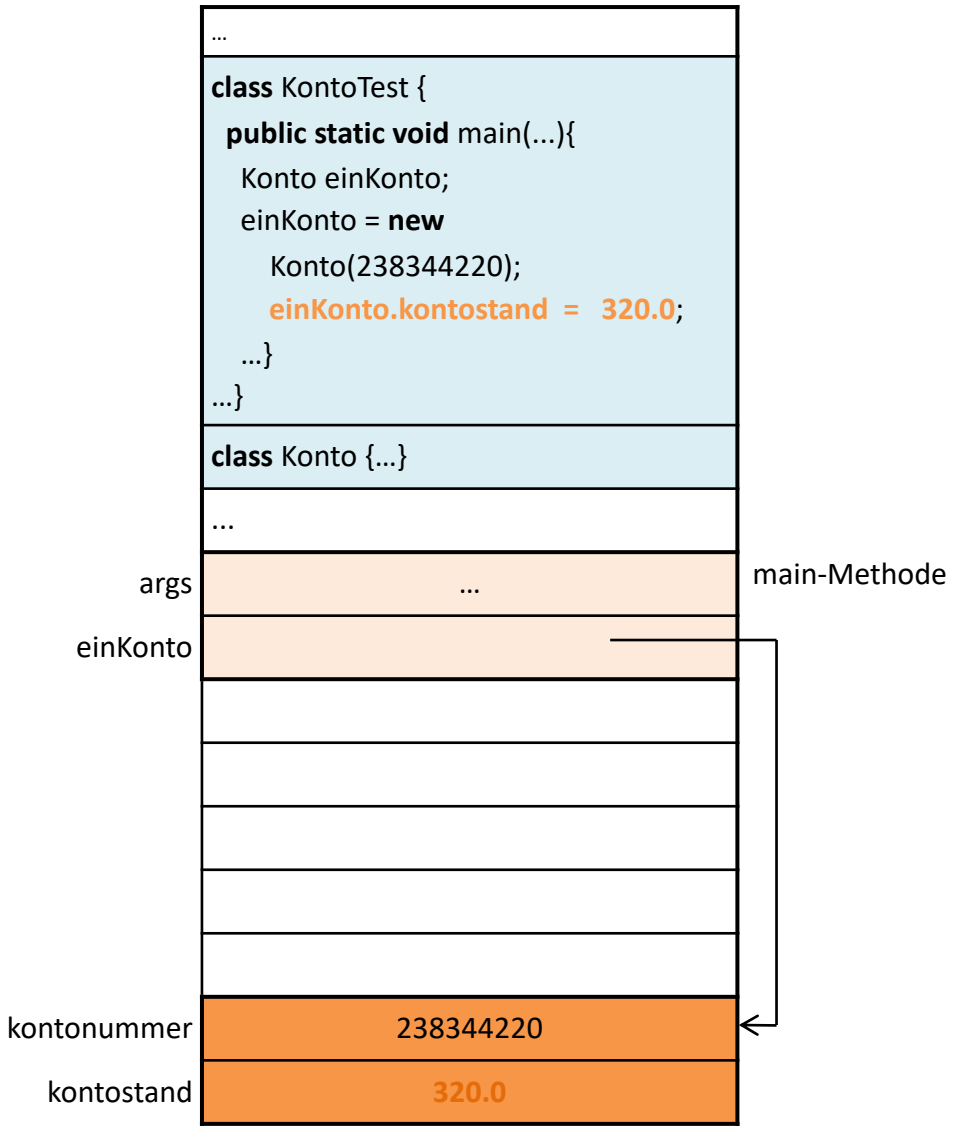
Beispiel: Schreibender Zugriff



Objekte in Java

Zugriff auf Objektattribute

Beispiel: Schreibender Zugriff



Objekte in Java

Aufruf einer Objektmethode in Java

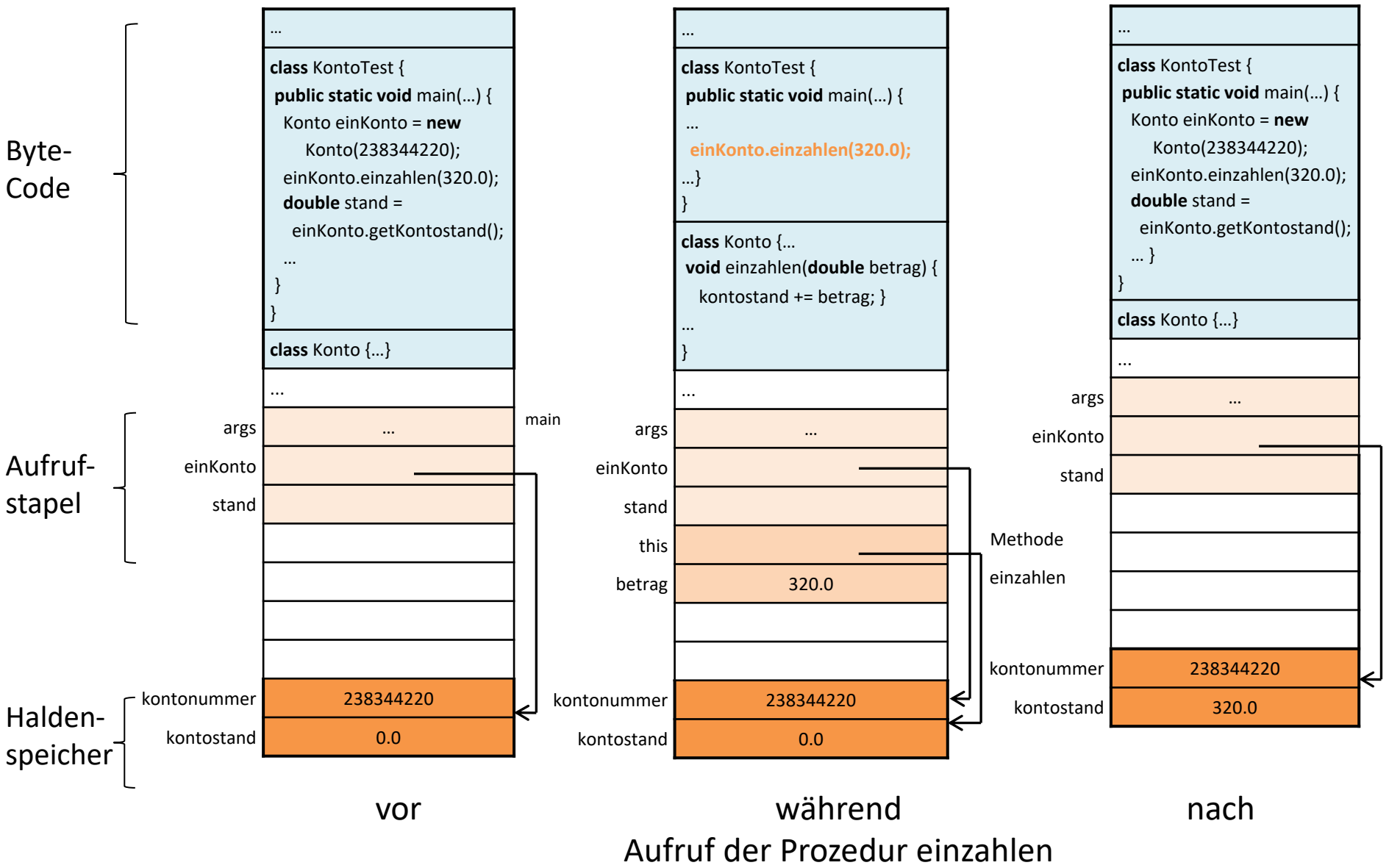
- Notation: `Objektname.Methodenname`

- Beispiel:

```
class KontoTest
{
    public static void main(String[] args)
    {
        Konto einKonto = new Konto(238344220);

        // Schreibender Zugriff
        einKonto.einzahlen(320.0);

        // Lesender Zugriff
        double stand = einKonto.getKontostand();
        System.out.println("Kontostand = " + stand);
    }
}
```

Objekte in Java

Aufruf einer Objektmethode im Speicher (Prozedur)

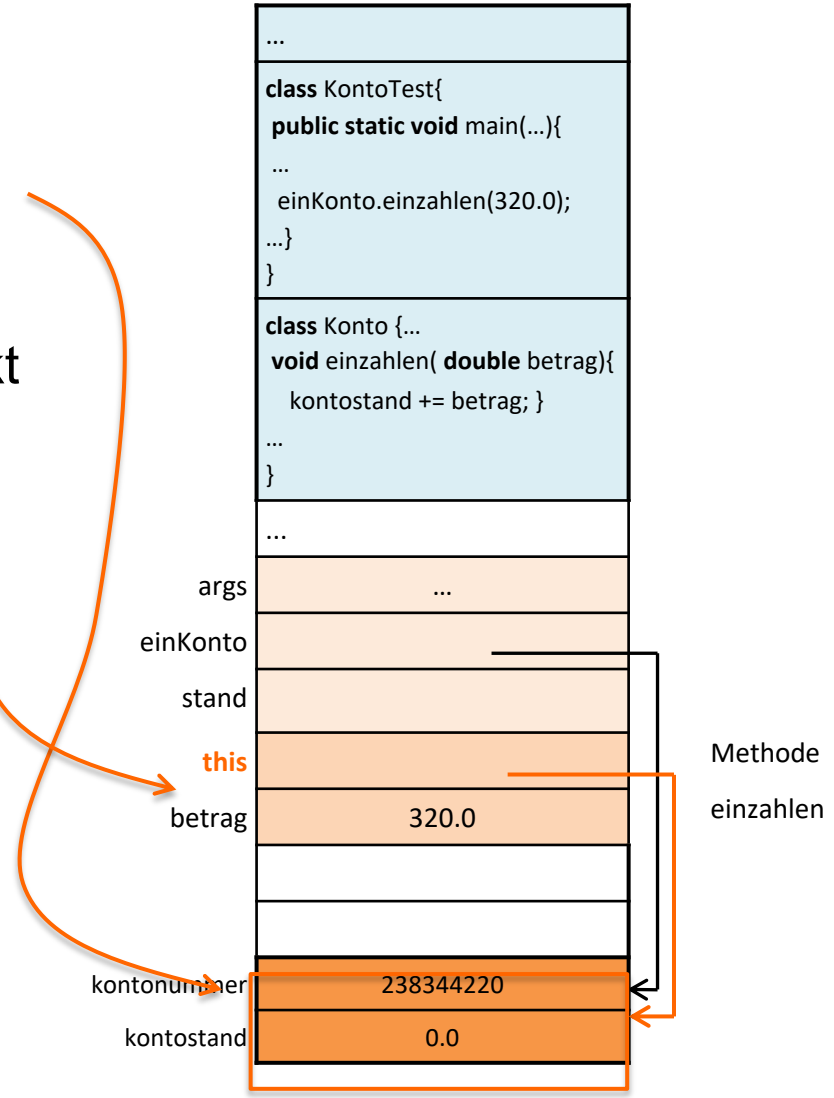


Objekte in Java

Aufruf einer Objektmethode im Speicher

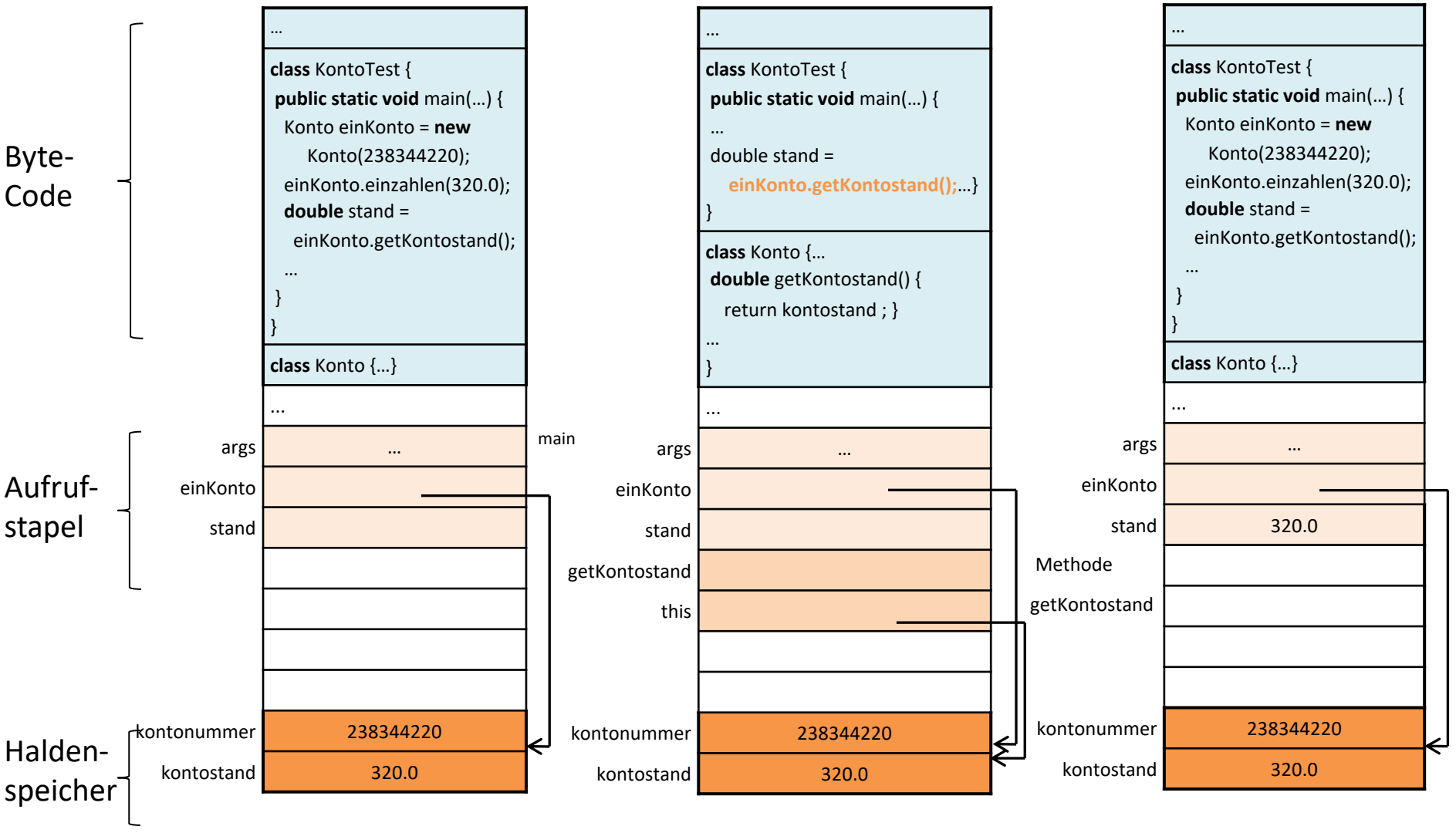
Objektmethode

- wird auf ein einzelnes Objekt einer Klasse angewendet
- enthält den versteckten Parameter `this` als Referenz auf dieses Objekt



Objekte in Java

Aufruf einer Objektmethode im Speicher (Funktion)



Aufruf der Funktion getKontostand

Lernziele

- ✓ Die Begriffe Objekte, Zustand, Verhalten, Klasse, Attribut und Methode erläutern können.
- ✓ Darstellung von Objekten und Klassen in UML lesen können.
- ✓ Eine Klasse in UML-Notation in Java-Klassen transformieren können.
- ✓ Referenzvariablen charakterisieren können
- ✓ Vorgänge bei der Erzeugung von Objekten und Zuweisung von Referenzen im Hauptspeicher skizzieren können
- ✓ Eine main-Methode in Java programmieren können, in der Objekte erzeugt, auf Attribute von Objekten zugegriffen und Methoden aufgerufen werden.

Literatur

- H. Balzert: Lehrbuch Grundlagen der Informatik, Spektrum Akademischer Verlag, 2. Auflage, 2004
- D. Ratz, J. Scheffler, D. Seese, J. Wiesenberger, Grundkurs Programmieren in JAVA, Hanser Verlag, 7. Auflage, 2014