CARL VON OSSIETZKY universität OLDENBURG

Teil

Objektorientierte Programmierung

Unterrichtseinheit 23

Definition von Klassen

Dr. Dietrich Boles

Gliederung

- Klasse
- Klassendefinition
  - Instanz-Attribute
  - Konstruktoren
  - Instanz-Methoden
- Beispiel Gewicht
- Definitionen

Klasse

```
<klassen-def> ::= ["public"] "class" <bezeichner> "{"
                    { <attr-def |
                      <konstruktor-def> |
                      <methoden-def> | ...
                    }
                  "}"
<attr-def>        ::= <variablen-def>
<methoden-def>    ::= <funktionen-def> (ohne static)
<konstruktor-def> ::= <methoden-def> (ohne Funktionstyp)
```

Anmerkungen:

- Bezeichner: **Klassenname** (neuer Typ!)
- **Klasse = Verbund inklusive Funktionen auf Verbund**
- Attribute: Gültigkeitsbereich ist der gesamte Klassenblock!
- Vereinbarung: Klassenname beginnt mit Großbuchstaben

Klassendefinition / Instanz-Attribute

- genau wie bei Verbunden
- Gültigkeitsbereich: der gesamte Klassenblock
- jedes Objekt einer Klasse hat dieselben Attribute
- die Wert der Attribute können jedoch verschieden sein!

```
class Hamster {

  // Instanz-Attribute

  int reihe;
  int spalte;
  int blickrichtung;
  int anzahlKoerner;
```

Klassendefinition / Konstruktoren

- spezielle Funktion/Methode zur Initialisierung der Attribute eines Objektes
- Funktionsname = Klassenname
- kein Funktionstyp (auch nicht `void`)
- wird bei der Erzeugung eines Objektes aufgerufen

```
// Konstruktoren

Hamster(int r, int s, int b, int k) {
  // Initialisierung aller Instanzattribute
  reihe = r;
  spalte = s;
  blickrichtung = b;
  anzahlKoerner = k;
}
```

Klassendefinition / Instanz-Methoden (1)

- wie Prozeduren / Funktionen, jedoch ohne `static`
- können auf Instanz-Attribute und (andere) Instanz-Methoden zugreifen

```
// Instanz-Methoden
void linksUm() {
  if (blickrichtung == Hamster.NORD) {
    blickrichtung = Hamster.WEST;
  } else if (blickrichtung == Hamster.WEST) {
    blickrichtung = Hamster.SUED;
  } else if (blickrichtung == Hamster.SUED) {
    blickrichtung = Hamster.OST;
  } else if (blickrichtung == Hamster.OST) {
    blickrichtung = Hamster.NORD;
  }
  // Aenderung auf dem Bildschirm sichtbar machen
}
```

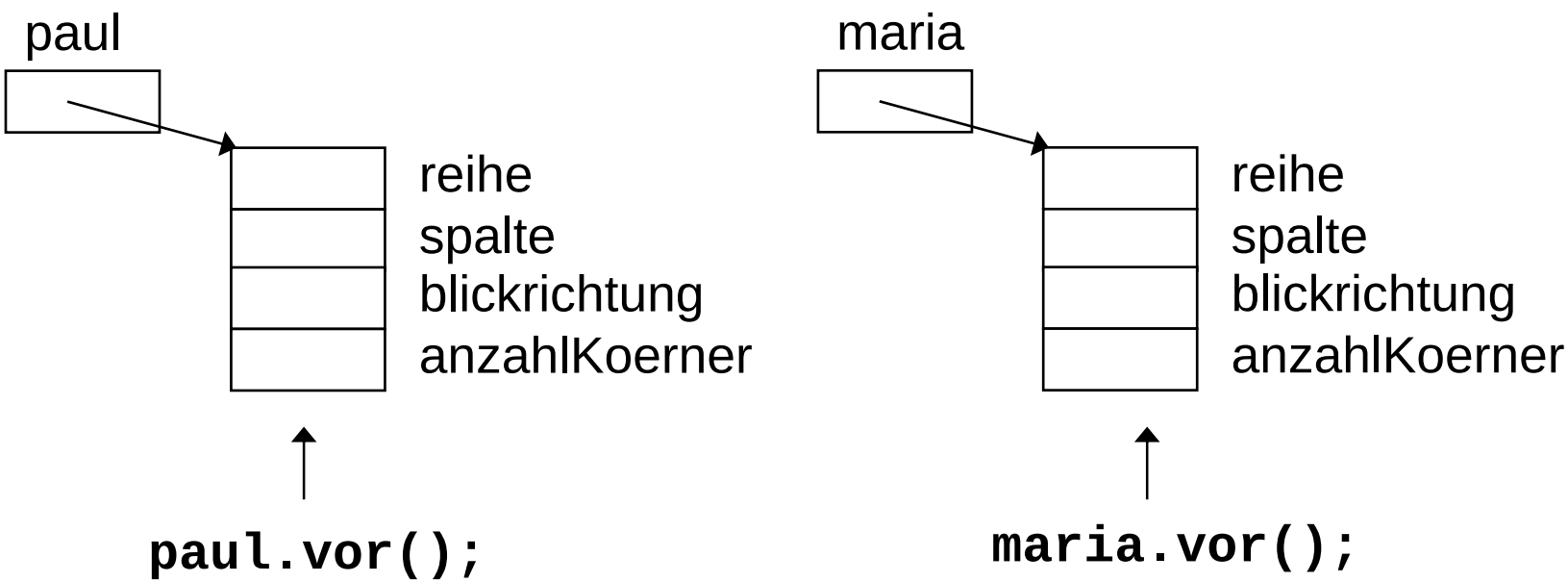
Klassendefinition / Instanz-Methoden (2)

```
  void vor() {
    if (!vornFrei()) { // oder: this.vornFrei()
      // Programmabbruch
    } else if (blickrichtung == Hamster.NORD) {
      reihe = reihe – 1;
    } else if (blickrichtung == Hamster.SUED) {
      reihe = reihe + 1;
    } else if (blickrichtung == Hamster.OST) {
      spalte = spalte + 1;
    } else if (blickrichtung == Hamster.WEST) {
      spalte = spalte – 1;
    }
    // Aenderung auf dem Bildschirm sichtbar machen
  }

  . . .
}
```

Klassendefinition / Objektrealisierung

```
Hamster paul = new Hamster(1,2,Hamster.OST,3);
Hamster maria = new Hamster(3,4,Hamster.NORD,5);
paul.vor();
maria.vor();
paul.linksUm();
maria.linksUm();
```



Beispiel Gewicht

Problemstellung:

Das so genannte Normalgewicht berechnet sich nach der Formel "Körpergröße (in cm) minus 100". Das Idealgewicht beträgt bei Männern 90% und bei Frauen 85% des Normalgewichts.

Schreiben Sie ein (objektorientiertes) Java-Programm, welches nach Eingabe von Größe, Gewicht und Geschlecht ausgibt, ob ein Mensch zu dick oder zu dünn ist, oder ob er/sie zwischen Ideal- und Normalgewicht liegt.

Demo

Beispiel Gewicht / Protokoll

```
class Mensch {

  Mensch(boolean maennlich,
         int groesse,  // in cm
         float gewicht // in kg
        )

  boolean hatUebergewicht()
  boolean hatUntergewicht()
  boolean hatOrdentlichesGewicht()

  void neueGroesse(int groesse)
  void neuesGewicht(float gewicht)

}
```

Beispiel Gewicht / Hauptprogramm

```
class OOGewicht {
  public static void main(String[] args) {
    Mensch person =
      new Mensch(IO.readChar("Maennlich (m/w)?") == 'm',
                 IO.readInt("Groesse (cm) eingeben: "),
                 IO.readFloat("Gewicht (kg) eingeben: "));

    if (person.hatUebergewicht()) {
      IO.println("Alter Fettsack!");
    } else if (person.hatUntergewicht()) {
      IO.println("Na, du Hungerhaken!");
    } else {
      IO.println("Idealer Gewichtsbereich!");
    }
  }
}
```

Beispiel Gewicht / Klassendefinition (1)

```
class Mensch {

  // Attribute
  boolean maennlich;
  int groesse; // in cm
  float gewicht; // in kg

  // Konstruktor
  Mensch(boolean mann, int groe, float gew) {
    maennlich = mann;
    groesse = groe;
    gewicht = gew;
  }
```

Beispiel Gewicht / Klassendefinition (2)

```
// Methoden
void neueGroesse(int groe) {
  groesse = groe;
}

void neuesGewicht(float gew) {
  gewicht = gew;
}

boolean hatUebergewicht() {
  return gewicht > berechneNormalgewicht();
}

boolean hatUntergewicht() {
  return gewicht < berechneIdealgewicht();
}

boolean hatOrdentlichesGewicht() {
  return gewicht <= berechneNormalgewicht() &&
         gewicht >= berechneIdealgewicht();
}
```

Beispiel Gewicht / Klassendefinition (3)

```
  // Hilfsmethoden

  float berechneNormalgewicht() {
    return groesse - 100;
  }

  float berechneIdealgewicht() {
    if (maennlich) {
      return berechneNormalgewicht() / 100.0f * 90.0f;
    } else {
      return berechneNormalgewicht() / 100.0f * 85.0f;
    }
  }
}
```

Definitionen (1)

Klasse:

beschreibt

- Eigenschaften (Attribute)
- Struktur (Subobjekte)
- Verhalten (Methoden)

einer Gruppe von gleichartigen Objekten ($\rightarrow$ Datentyp)

Objekte (**Instanzen**):

- werden durch Klassen beschrieben
- setzen sich demnach zusammen aus
  - Datenelementen (Attribute) $\rightarrow$ Eigenschaften / Struktur / Zustand
  - den auf den Attributen ausführbaren Operationen $\rightarrow$ Verhalten
- Objekte einer Klasse haben gleiche Attribute und gleiche Funktionen; sie unterscheiden sich nur in den Werten ihrer Attribute

Definitionen (2)

Attribut (Instanz-Variable):

- Variable, für die in jedem Objekt Speicherplatz reserviert ist
- Menge der Attribute eines Objektes repräsentiert Zustand eines Objektes
- ($\rightarrow$ Attribut eines Verbundes)

Methode:

- realisieren die auf Objekten einer Klasse anwendbaren Operationen
- ($\rightarrow$ Funktionen auf Verbund (d.h. Attributen))

Konstruktor:

- spezielle Methode zur Initialisierung von Objekten

Instantiierung:

- Erzeugung von Objekten