CARL VON OSSIETZKY universität OLDENBURG

Teil

Imperative Programmierung

Unterrichtseinheit 19

Verbunde

Dr. Dietrich Boles

Gliederung

- Motivation
- Verbundtypen
- Verbundvariablen
- Verbunderzeugung
- Verbundattribute
- Arrays mit Verbunden
- Arrays und Verbunde als Attribute
- Beispiele
- „Abstrakter Datentyp“
- Zusammenfassung

Motivation (1)

Definition "Verbund":

Datenstruktur/-typ zur Zusammenfassung von mehreren **Elementen u.U. unterschiedlichen Typs** zu einer Einheit. Der Wertebereich eines Verbundes ist das kartesische Produkt der Wertebereiche seiner einzelnen Elemente (auch **Attribute** genannt)

Synomyme:

Records, Strukturen, Datensätze

Anmerkungen:

- Verbunde gibt es in Java nicht; man kann sie aber mit Hilfe von Klassen bzw. Objekten nachbilden
- Verbundtypen sind Referenzdatentypen!

Motivation (2)

Beispiel: Personaldatei

Mitarbeiter:

Name:	String name;
Alter:	short alter;
Geschlecht:	boolean maennlich;
Gehalt:	float gehalt;

Java:

Definition des Verbundtyps:

```
class Mitarbeiter {
  String name;
  short alter;
  boolean maennlich;
  float gehalt = 2000.0F;
}
```

Definition einer Verbundvariablen:

```
Mitarbeiter karl;
```

Nutzung der Variablen:

```
karl.alter = 42;
```

Verbundtypen / Definition

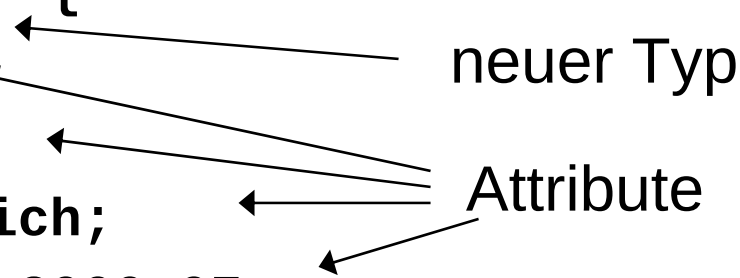
```
<Verbundtyp> ::= "class" <Bezeichner>
                 "{" { <Variablendefinition> } "}"
```

Anmerkungen:

- der Bezeichner legt den Namen für den neuen Typ fest
- der neue Typ kann überall dort stehen, wo auch Standarddatentypen stehen dürfen
- die Variablen nennt man auch Elemente oder Attribute

Beispiel: Datei **Personalverwaltung.java**

```
class Mitarbeiter {
  String name;
  short alter;
  boolean maennlich;
  float gehalt = 2000.0F;
}
class Personalverwaltung {...}
```



Verbundvariablen / Definition

```
<Verbundvar-Def> ::= <Verbundtyp> <Bezeichner>
                     {"," <Bezeichner> } ";"
```

Anmerkungen:

- <Verbundtyp> muss gültiger Verbundtyp sein
- Reservierung von Speicherplatz für Adressen (Referenzen auf Verbünde)
- <Bezeichner> ist der Name der Verbundvariablen
- kann Referenzen (Adressen) auf Verbunde speichern

Beispiele:

```
Mitarbeiter karl;

Mitarbeiter hans, hubert;

Mitarbeiter heike = null; // Initialisierung
```

Verbunderzeugung

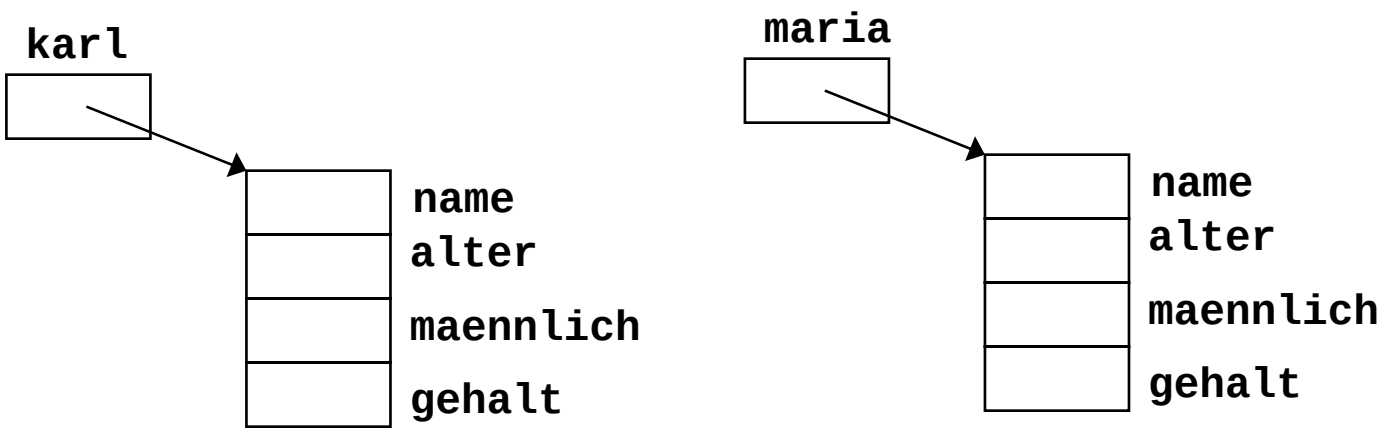
<Verbund-Erz> ::= "new" <Verbundtyp> "(" ")"

Anmerkungen:

- <Verbundtyp> muss gültiger Verbundtyp sein
- Reservierung von Speicherplatz für Verbundattribute (auf dem Heap)
- liefert Referenz (Adresse) des Speicherbereiches

Beispiele:

```
Mitarbeiter karl = new Mitarbeiter();
Mitarbeiter maria = new Mitarbeiter();
```



Verbundattribute / Zugriff

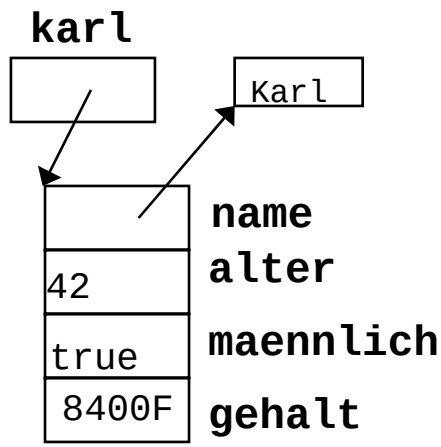
```
<Verbund-Attr-Zugr> ::= <verbundvar-bezeichner>
                        "."
                        <verbund-attr>
```

Anmerkungen:

- <verbundvar-bezeichner> muss gültiger Name einer Verbundvariablen sein
- <verbund-attr> muss Attribut eines Verbundes vom Typ der Verbundvariablen sein
- Verbund muss vorher erzeugt worden sein (-> Laufzeitfehler!)
- können überall dort stehen, wo auch Variablennamen stehen dürfen

Beispiele:

```
Mitarbeiter karl = new Mitarbeiter();
karl.name = "Karl";
karl.alter = 42;
karl.maennlich = true;
karl.gehalt = 2000.0F * (karl.alter / 10.0F);
```

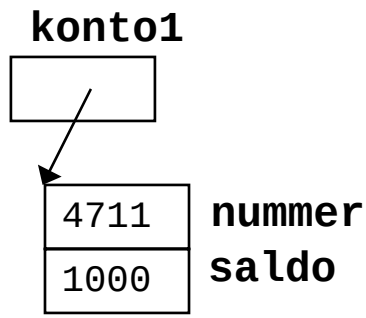


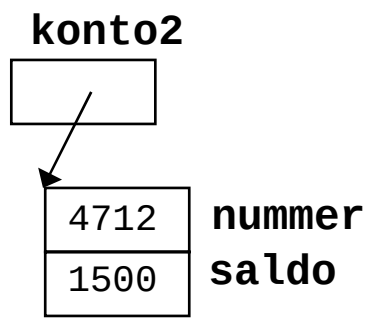
Verbundattribute / Initialisierung

- bei der Definition des Verbundtyps (allgemeingültig)
- durch Default-Werte des entsprechenden Typs
- explizit für jedes Attribut

Beispiele:

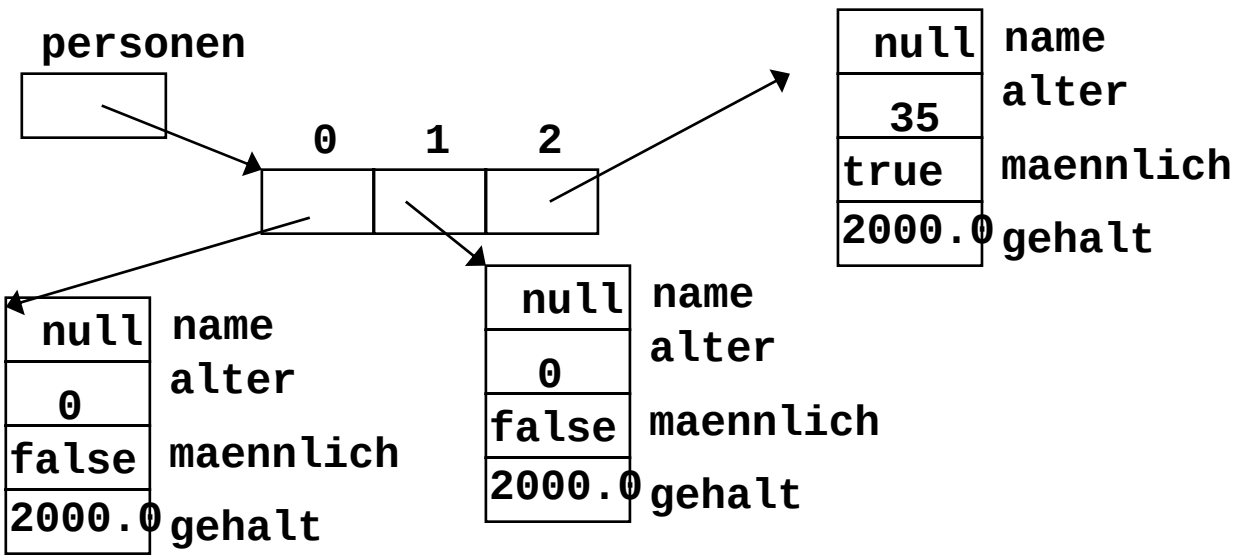
```
class Konto {
   int nummer;
   int saldo = 1000;
} ...
public static void main(String[] a){
  Konto konto1 = new Konto();
  konto1.nummer = 4711;
  Konto konto2 = new Konto();
  konto2.nummer = 4712;
  konto2.saldo += 500;
}
```





Arrays mit Verbunden

```
Mitarbeiter[] personen = new Mitarbeiter[3];
for (int i=0; i<3; i++) {
  personen[i] = new Mitarbeiter();
}
personen[2].alter = 35;
personen[2].maennlich = true;
```



Arrays und Verbunde als Attribute

```
class Produkt {
  String bez;
  float  preis;
}

class Kunde {
  int    nummer;
  String name;
}

class Warenkorb {
  Kunde     kaeufer;
  Produkt[] produkte;
}
```

```
public class Shop {

public static void main(String[] a) {

  Warenkorb korb = new Warenkorb();

  korb.kaeufer = new Kunde();
  korb.kaeufer.nummer = 4711;
  korb.kaeufer.name = "Karl";

  korb.produkte = new Produkt[3];

  korb.produkte[0] = new Produkt();
  korb.produkte[0].bez = "Hamster-Buch";
  korb.produkte[0].preis = 24.90F;
  ...
}}
```

Beispiel 1 (1)

```
class Position { int zeile; int spalte; }

class SchiebeSpiel {
  public static void main(String[] args) {
    int[][] matrix = { {2, 14, 5, 7},
                       {3, 4, 15, 13},
                       {6, 1, 12, 11},
                       {3, 8, 9, 0} };
    print(matrix);
    while (!korrekt(matrix)) {
      Position position = posEingabe(matrix);
      verschieben(matrix, position);
      print(matrix);
    }
  }
```

2	14	5	7
3	4	15	13
6	1	12	11
3	8	9	

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Demo

http://www.mazeworks.com/sliders/index.htm

Beispiel 1 (2)

```
static void print(int[][] matrix) {
  for (int i=0; i<matrix.length; i++) {
    for (int j=0; j<matrix[i].length; j++) {
      if (matrix[i][j] <= 10)
        IO.print(" ");
      IO.print(matrix[i][j] + " ");
    }
    IO.println();
  }
  IO.println();
}
```

Beispiel 1 (3)

```
static boolean korrekt(int[][] matrix) {
  int vgl = 0;
  for (int i=0; i<matrix.length; i++) {
    for (int j=0; j<matrix[i].length; j++) {
      if (vgl != matrix[i][j]) return false;
      vgl++;
  } }
  return true;
}

static Position posEingabe(int[][] matrix) {
  // Achtung: erwartet eine korrekte Eingabe!
  Position pos = new Position();
  pos.zeile = IO.readInt("korrekte Zeile eingeben: ");
  pos.spalte = IO.readInt("korrekte Spalte eingeben: ");
  return pos;
}
```

Beispiel 1 (4)

```
  static void verschieben(int[][] m, Position pos) {
    Position frei = freiesFeld(m);
    m[frei.zeile][frei.spalte] = m[pos.zeile][pos.spalte];
    m[pos.zeile][pos.spalte] = 0;
  }
  static Position freiesFeld(int[][] matrix) {
    for (int i=0; i<matrix.length; i++) {
      for (int j=0; j<matrix[i].length; j++) {
        if (matrix[i][j] == 0) {
          Position pos = new Position();
          pos.zeile = i; pos.spalte = j;
          return pos;
    } } }
    return null; // sollte eigentlich nicht vorkommen!
  }
}  // end class
```

Beispiel 2 (1)

```
class Bruch {
  int zaehler = 1;
  int nenner = 1;
}
```

Abstrakter Datentyp (ADT) =

Datenstruktur

+

Funktionen auf der Datenstruktur

```
class BruchRechnung {

  static void init(Bruch bruch, int z, int n) {
    bruch.zaehler = z;
    bruch.nenner = n;
    kuerzen(bruch);
  }
```

Beispiel 2 (2)

```
static void multTo(Bruch b1, Bruch b2) {
  b1.zaehler *= b2.zaehler;
  b1.nenner *= b2.nenner;
  kuerzen(b1);
}


static void addTo(Bruch b1, Bruch b2) {
  b1.zaehler =
    b2.nenner * b1.zaehler +
    b1.nenner * b2.zaehler;
  b1.nenner = b1.nenner * b2.nenner;
  kuerzen(b1);
}
```

Beispiel 2 (3)

```
static String toString(Bruch b) {
  return b.zaehler + "/" + b.nenner;
}

static void kuerzen(Bruch b) {
  int ggt = ggT(b.zaehler, b.nenner);
  b.zaehler /= ggt;
  b.nenner /= ggt;
}

static int ggT(int z1, int z2) {
  if (z2 == 0) {
    return z1;
  } else {
    return ggT(z2, z1 % z2);
  }
}
```

Beispiel 2 (4)

```
  public static void main(String[] args) {
     int zaehler = IO.readInt("Bruch 1 (Zaehler): ");
     int nenner = IO.readInt("Bruch 1 (Nenner): ");
     Bruch b1 = new Bruch();
     init(b1, zaehler, nenner);
     IO.println("Bruch 1: " + toString(b1));

     zaehler = IO.readInt("Bruch 2 (Zaehler): ");
     nenner = IO.readInt("Bruch 2 (Nenner): ");
     Bruch b2 = new Bruch();
     init(b2, zaehler, nenner);
     IO.println("Bruch 2: " + toString(b2));

     multTo(b1, b2);
     IO.println("Bruch 1: " + toString(b1));
     IO.println("Bruch 2: " + toString(b2));

     addTo(b1, b2);
     IO.println("Bruch 1: " + toString(b1));
     IO.println("Bruch 2: " + toString(b2));
} }
```

Zusammenfassung

- Verbund: Zusammenfassung mehrerer Variablen u. U. unterschiedlichen Typs zu einer Einheit
- Verbundtyp: benutzerdefinierter Datentyp
- Verbundtypen sind Referenzdatentypen
- Verbundvariable: Variable zur Referenzierung eines Verbundes
- Zugriff auf die einzelnen Variablen (Attribute) eines Verbundes via Verbundvariable und Punktnotation