Felder - Arrays

Variable gleichen Types können in Feldern (array) zusammengefasst werden.

Typ[] feldname; oder Typ feldname[];

dabei kann unter Benutzung des new-Operators gleich die Dimension zugewiesen werden

Typ feldname[] = new Typ[<ganze Zahl >];

Beispiel: double vektor[] = new double[5];

auch eine Initialisierung ist möglich.

Beispiel: double vektor[] = {1.9, 1e-6, 9.9};

Benutzung von Feldern, mehrdimensionale Felder

- Um auf ein einzelnes Feldelement zuzugreifen wird feldname[<index>] geschrieben.

  <index> ist dabei ein ganzzahliger Ausdruck vom Typ int.

  Beispiel: feldname[4]
- Der Index beginnt stets bei 0.
- Jedem Feld ist die Variable length zugeordnet, die mit feldname.length abgefragt werden kann.
- Mehrdimensionale Felder (z.B. Matrizen, Tensoren) werden durch Felder von Feldern realisiert.

  Beispiel: double feldname[][] = new double[3][7];

Kopieren von Feldern

Sollen die Inhalte eines Feldes feld_1 in ein Feld feld_2 kopiert werden gibt es zwei Möglichkeiten:

1. Es wird elementweise gemacht
   for(int i=0; i < feld_1.length; i++) feld_2[i] = feld_1[i];
2. Es wird die Systemroutine System.arraycopy benutzt:
   arraycopy(Object quelle, int quellindex, Object ziel, int zielindex, int laenge);
   Beispiel: System.arraycopy(feld_1, 0, feld_2, 0, feld_1.length);

Die Variante feld_2 = feld_1; ist falsch, weil damit das Objekt feld_2 mit dem Objekt feld_1 identifiziert wird.
Beide sind jetzt die selben Objekte.
Ändere ich feld_1, so ändert sich automatisch feld_2 mit.

Die Welt besteht aus Objekten, Klassen beschreiben Objekte.
Objekte sind klassifizierbar, haben Eigenschaften.

Beispiel: Die Klasse (Menge) der Computer lässt sich beschreiben durch:
Betriebssystem: Windoof, linux, Mac
Standort: Zimmernummer
Zustand: eingeschaltet, ausgeschaltet

Objekte sind konkrete Mitglieder, Realisierungen, Instanzen einer Klasse.
Methoden verändern die Eigenschaften oder benutzen sie.

```
public class <klassenname> {

}
```

modifikator_1 modifikator_2 class <klassenname>{

modifikator_1: leer oder public
modifikator_2: leer oder abstract oder final

}

public class <klassenname> {

Klassenvariable; statische (globale) Variable, die für alle Objekte gleich sind.

Instanzvariable; beinhalten die Eigenschaften jedes Objektes

Konstruktor(en); Methoden zum Erzeugen eines Objektes der Klasse

Instanzmethode(n);

statische Methode(n); Klassische Unterprogramme. Dürfen Klassenvariable benutzen, aber keine Instanzvariablen.

}

Methoden

```
public static <typ> <name>(<Parameterliste>){
  ⋮                          Nutzung von Klassenvariablen möglich
return Variable vom Typ <typ>;
}
```

oder

```
public <typ> <name>(<Parameterliste>){
  ⋮                    Nutzung von Instanzvariablen und Klassenvariablen möglich
return Variable vom Typ <typ>;
}
```

-Parameterliste: enthält alle Daten, die notwendig sind. Aufzählung von <typ> <variablenName> durch Kommata getrennt.

-Typ void (leer), falls nichts zurückgegeben wird.

Konstruktoren

- dienen der Erzeugung von Objekten. Sie müssen die Eigenschaften des Objektes festlegen.
- Konstruktorname = Klassenname
- Sie haben keinen Typ (sie sind selbst einer) und keinen Rückgabewert (kein return)
- Gibt es mehrere Konstruktoren, so müssen sie sich in der Parameterliste unterscheiden.
- Gibt es keinen Konstruktor, so wird automatisch der triviale Konstruktor angelegt. Klassenname()

```
public class Computer {
.       public String betriebssystem;// windows, linux, mac, unbekannt
.       public int zimmer; .   .       // Haus-Etage-Zimmer ohne Punkt
.       public boolean zustand; .      // true angeschaltet, false abgeschaltet
.
.       public Computer(String bs, int wo, boolean on_off){
.       .        betriebssystem=bs;
.       .        zimmer=wo;
.       .        zustand=on_off;
.       }
.       public Computer change_zustand(){
.       .       this.zustand=!zustand;
.       .       return this;
.       }
.       public Computer umzug(int neuesZimmer){
.       .       this.zimmer=neuesZimmer;
.       .       return this;
.       }
.       public int welchesZimmer(){
.       .       return this.zimmer;
.       }
.       public static void Ausgabe(Computer x){
.       .       String zustand;
.       .       if (x.zustand)
.       .       .       zustand="angeschaltet";
.       .       else
.       .       .       zustand="abgeschaltet";
.       .       System.out.println("Der Computer l\u00e4uft mit "
.       .       .       +x.betriebssystem+", ist "+zustand+
.       .       .       " und steht im Zimmer "+x.zimmer);
.       }
}
```

Objekte und Instanzen

Anlegen eines Objektes einer Klasse:

<klName> <varName> = new $\underbrace{\text{< klName > (< ParaListe >);}}_{\text{Konstruktor der Klasse}}$

Nutzen von Instanzmethoden bzw. Instanzvariablen (nicht static):

<varName>.<methodenName>(ParaListe) bzw.
<varName>.<instanzVarName>

All Classes
Computer

Class Computer

java.lang.Object
└**Computer**

public class **Computer**
extends java.lang.Object

Field Summary

java.lang.String	**betriebssystem**
int	**zimmer**
boolean	**zustand**

Constructor Summary

Computer(java.lang.String bs, int wo, boolean on_off)

Method Summary

static void	**Ausgabe**(Computer x)
Computer	**change_zustand**()
Computer	**umzug**(int neuesZimmer)
int	**welchesZimmer**()

```
public class ComputerTest{
        public static void main(String[] args){

                Computer meiner=new Computer("linux",2415,true);

                Computer imPool;

                Computer.Ausgabe(meiner);

        // Rechner ausschalten

                meiner=meiner.change_zustand();

                Computer.Ausgabe(meiner);

                imPool=meiner.umzug(2207);

                System.out.println("Mein Rechner steht jetzt im Raum "+imPool.zimmer);
        }
}
```

```
java ComputerTest

        Der Computer läuft mit linux, ist angeschaltet und steht im Zimmer 2415

        Der Computer läuft mit linux, ist abgeschaltet und steht im Zimmer 2415

        Mein Rechner steht jetzt im Raum 2207
```

Applikationen

Nun verstehen wir auch das main-Programm:

```
public static void main (Strings[] arg){
}
```

Java erwartet genau diese Spezifikation zum Ausführen einer Applikation!

Das Feld von Strings namens arg, sind Argumente, die beim Aufruf des Java-Programmes verwendet werden können.

Bsp.: java test 45 "x" 2.0

Dann ist: arg[0]="45", arg[1]="x", arg[2]="2.0"

Implizite Typumwandluungen

bei primitiven Datentypen, die problemlos sind, werden oft implizit durchgeführt.

Beispiel: int i=7; double x=i;

Bei der Darstellung von Zeichenketten wird die Methode toString() verwendet.

Daher funktioniert System.out.println(" x= "+x);

Hier wird x ersetzt durch Double.toString(x)

Für jedes (ordentliche) Objekt gibt es eine toString()-Methode.

Strings

sind Zeichenketten, die aus einzelnen Symbolen vom Typ char bestehen.

Sie sind wie in einem Feld angeordnet, beginnend mit dem Index 0.

Bsp.: String z="Bald ist Weihnachten";

char buchstabe = z.charAt(10); liefert ?

Zum Vergleich von Zeichenketten benutzt man compareTo(String z)