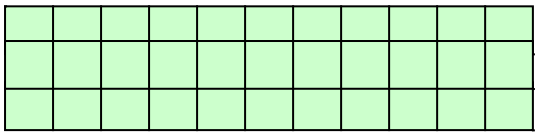
Statische Arrays

Ein Array ist ein Behälter für Datenelemente desselben Typs. Arrays können beliebig viele Dimensionen haben. Die Standardform ist das eindimensionale Array, das man sich in Zeilenform vorstellen kann. Ein eindimensionales Array wird auch Vektor genannt.

eindimensionales Array (Vektor) [x]

Bei mehrdimensionalen Arrays müssen alle Zeilen einer Dimension die selbe Länge haben. Die Elemente aller Dimensionen müssen vom selben Datentyp sein.



zweidimensionales Array [y] [x]

Ein statisches Array ist ein Objekt der Klasse `java.lang.reflect.Array`, gleichgültig, aus welchem Datentyp die einzelnen Elemente bestehen. Diese Klasse braucht bei der Erzeugung eines Arrays aber nicht genannt werden, da der Compiler die Arrayeigenschaft an den **eckigen Klammern** erkennt.

Bei der Erzeugung eines Arrays muß die maximale Anzahl der Elemente (die Länge des Arrays) angegeben werden. In Java kann die Länge eines Arrays zur Laufzeit festgelegt werden – zum Beispiel durch Benutzereingabe.

Statische Arrays können prinzipiell alle Arten von Datentypen aufnehmen:

- Wertdatentypen (elementare Datentypen - int, float, char)
- Objektdatentypen (selbstdefinierte Objekte, API-Objekte)

Die Elemente eines Arrays sind durchnummeriert – das erste Element hat den Index [0], das letzte Element den Index [Länge-1]. Bei mehrdimensionalen Arrays wird die folgende Dimension jeweils links davor erwähnt.

Array [Dimension 2] [Dimension 1]

Erzeugung eines statischen Arrays

Ein Array besteht (wie alle Java-Objekte) aus einem Namen (der Referenzvariable) und dem eigentlichen Objekt, das durch den speziellen Konstruktoraufruf mit Angabe der Array-Länge erzeugt wird. Der Erzeugungsprozeß besteht deshalb auch aus mehreren Teilen: der Datentyp des Arrays wird durch Anhängen von **leeren eckigen Klammern** festgelegt; der Konstruktoraufruf hat im Unterschied zu allen anderen Java-Objekten **eckige** statt runde **Klammern** und enthält die Anzahl der Elemente:

1. Festlegung des Datentyps der Elemente des Arrays,
2. Arraykennzeichen: **[** leere eckige Klammern **]**,
3. Namen des Arrays,
4. Zuweisungsoperator,
5. Erzeugung des Arrayobjekts mit new und **[** Anzahl der Elemente **]**

Es gibt drei Möglichkeiten, ein Array zu erzeigen:

1.) Deklaration und Definition in einer Zeile

```
int [] einVektor = new int [10];
```

2.) verzögerte Erzeugung (in zwei Zeilen)

```
int [] einVektor;
int  einVektor [];    //alternativ

einVektor = new int [10];
```

3.) Initialisierung ohne explizite Längenangabe

```
int [] einVektor = { 3,5,4,2,78,2 };
```

Elementzugriff

Jedes Element eines statischen Arrays wird durch seine Position (den Abstand zum ersten Element) identifiziert, die durch den nullbasierten Index ausgedrückt wird. Auf Elemente des Arrays wird mittels des Index (in eckigen Klammern) zugegriffen. Es gibt keine Zugriffsmethoden (wie bei dynamischen Arrays), sondern es genügt der einfache Zuweisungsoperator **=**.

```
einVektor[4] = 3; // Element-Zugriff
```

```
int laenge = 30;
double[] doubleVektor;              // Name des Vektors
doubleVektor = new double[laenge];  // Vektorobjekt erzeugen


for(int i = 0; i < doubleVektor.length; i++)
   doubleVektor[i] = i;  // Elemente einlesen
```

Länge des Arrays

Die int-Variable `length` ist Attribut eines jeden statischen Arrays und gibt die maximale Anzahl der Elemente an.

Wenn versucht wird, auf Array-Elemente zuzugreifen, die außerhalb der deklarierten Bereichsgrenzen liegen, meldet der Java-Compiler keinen Fehler. Erst zur Laufzeit meldet der Java-Interpreter einen Fehler.

```
java.lang.ArrayIndexOutOfBoundsException at....
```

```
class vektorTest {

public static void main(String [] a) {

    int [] vektor;                // Statischer Vektor vom Typ int
    vektor = new int[10];         // 10 Elemente

    for (int i=0; i < 15; i++)
        System.out.println(vektor[i]);  // Exception bei vektor[10]

    }
}
```

Die Basisklasse **java.lang.reflect.Array** bietet einige nützliche Methoden, die zum Auslesen und Manipulieren der Elemente eines Arrays eingesetzt werden können:

```
import java.lang.reflect.Array;

class vektorTest {

public static void main(String [] a) {

    int [] vektor = new int[10];

    System.out.println("Der Vektor hat Platz fuer "
                       +  Array.getLength(vektor) +
                       " Elemente");
    }
}
```

Auch die Klasse **java.util.Arrays** bietet weitere nützliche Methoden, die auf Arrays angewendet werden können. Es gibt z.B. Methoden zum Suchen nach Elementen und Sortieren des ganzen Arrays:

```
import java.util.*;

class vektorSort {

  public static void main(String [] a) {

    int vektor[] = {2,5,1,7,5,6,8,0};
    Arrays.sort(vektor);
    for(int i=0; i < vektor.length; i++)
        System.out.println(vektor[i]);  // 0 1 2 3 4 5 6 7 8

     }
}
```

Statische Arrays aus Objekten

Wie bereits erwähnt, können statische Arrays nicht nur elementare Wertdatentypen (int, float, char...) aufnehmen, sondern auch Objekte jeglicher Art (auch selbstdefinierte). Objekt-Arrays werden in Programmen oft eingesetzt, um viele Objekte auf einfache Weise zu verwalten. Damit wird vermieden, daß jedes einzelne Objekt einen eigenen Namen besitzen muß und das Element kann einfach über seinen Index angesprochen werden.

BEISPIEL

```
//////////////////////////////////////////////////////
// Objekte werden in einem dynamischen Vektor erzeugt

class Student02 {

     int    matrikelnr;   // Attribute
     double einkommen;
     String vorname, nachname;

     public Student02( ) {  // KONSTRUKTOR:

        this.matrikelnr = 1;
        this.einkommen  = 2000;
        this.vorname    = "default";
        this.nachname   = "default";
        System.out.println("Konstruktor aufgerufen...");
     }

     public void gibAus() {  // Objekt-Attribute ausgeben

        System.out.println("\nStudent:"+vorname + " " + nachname);
        System.out.println("Einkommen: " + einkommen);
        System.out.println("MatrikelNr:" + matrikelnr);
     }

}
```

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

          System.out.print("Wieviele Studenten ?");
          int anzahl = In.readInt();

          Student02[] vektor = new Student02 [anzahl];

          for(int i=0; i < vektor.length; i++)
            vektor[i] = new Student02();  // Objekt

          for(int i=0; i < vektor.length; i++)
                vektor[i].gibAus();
    }
```