Informatik: Arrays

Ziele

- Verstehen wie ein Array deklariert, erstellt und verwendet wird
- Die Anzahl der Elemente in einem Array bestimmen können
- Das Kopieren und vergleichen von Arrays beherrschen können
- Mit Arrays in Methoden arbeiten können, diese über Methoden manipulieren und neu erstellen können
- Mit geschachtelten, mehrdimensionalen Arrays umgehen können

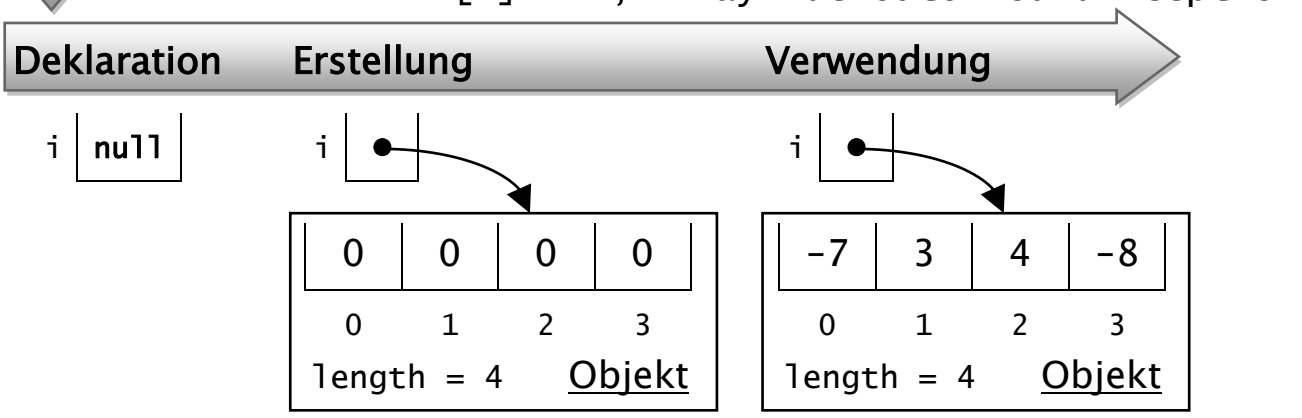
Arrays (dtsch. Felder)

- Speichern mehrere Elemente (primitive Datentypen, Objekte) gleichen Typs
- Zählen wie die Strings zu den Objekten
- Vorhandene Elemente können überschrieben werden
- Ist Array einmal angelegt kann die Anzahl der Elemente nicht geändert werden

Deklaration:
```
int[] i = null;
```

Erstellung (Instanziierung):
```
i = new int[4];
```

Verwendung:
```
i[0] = -7;
i[1] = 3;
i[2] = 4;
i[3] = -8;
i[4] = 1; ArrayIndexOutOfBoundException
```



Beispiel

```
boolean[] b = new boolean[100];
double[] d = {-7.3, 3.15, 4.6, 8};

double summe = 0.0;
for (int i = 0; i < d.length; i = i + 1) {
   double x = d[i];
   summe = summe + x;
}
```

`length` ist eine Variable des Objektes

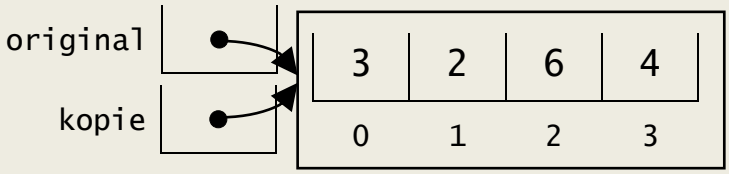
oder

```
for(double x: d)
   summe = summe + x;
```

- Beim Instanziieren werden den Arrayelementen Defaultwerte zugewiesen: **`false`**, **`null`**, 0, \u0000
- Der *Index* im Array muss vom Typ **`int`** sein: `d[3.5] = 7;`

Beispiel: Arrays kopieren und vergleichen

```
int[] original =
   {3, 2, 6, 4};
int[] kopie = original;
original[0] = 1;
System.out.println(kopie[0]);
```



Wertzuweisung an ein Array kopiert nur den Zeiger (Referenz)

```
kopie = original.clone();
```

Durch `clone()` wird eine vollständige Kopie des Originals erzeugt (funktioniert nur bei Elementen primitiven Datentyps)

```
if (original.equals(kopie)) …                    // ergibt false!!!
```

Durch `equals()` werden – zum Unterschied zu Strings – nur Objektvariablen und nicht die Arrayinhalte verglichen

```
kopie[0] = -1;
if (original[0] != kopie[0]) …                   // ergibt true!!!

boolean equal = (original.length == kopie.length);
int i = 0;
while (equal && i < original.length) {
   equal = (original[i] == kopie[i]);
   i = i + 1;
}
```

Im letzten Teil werden die Inhalte zweier Arrays elementweise verglichen

Beispiel: Array als Parameter

```
char[] c = new char[12];
fuellen(c, "Hallo");
for (int i = 0; i < c.length; i = i + 1)
   System.out.print(c[i]);                    // ergibt Hallo□□□□□□□
c = null;
erstellenUndFuellen(c, "Hallo Rosi");
if (c == null) …                              // ergibt true
c = erstellenUndFuellen("Hallo Edi");
for (int i = 0; i < c.length; i = i + 1)
   System.out.print(c[i]);                    // ergibt Hallo Edi□
```

```
public static void fuellen(char[] c, String s) {
   for (int i = 0; i < s.length(); i = i + 1)
      c[i] = s.charAt(i);
}
```

Zeiger auf Array wird als Kopie übergeben, über `c[i]` wird nicht Zeiger sondern Inhalt des Arrays geändert

```
public static void erstellenUndFuellen(char[] c, String s) {
   c = new char[10];
   for (int i = 0; i < s.length(); i = i + 1)
      c[i] = s.charAt(i);
}
```

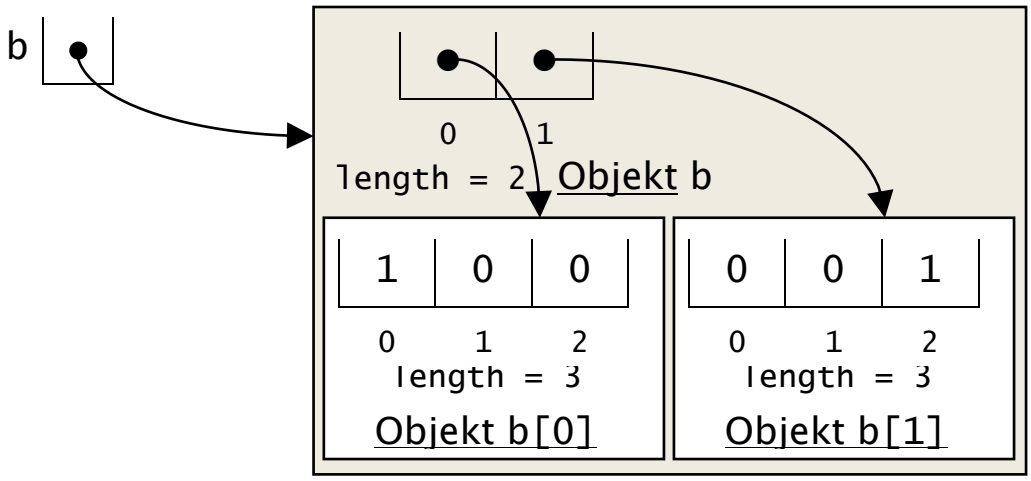
Zeiger auf Array wird als Kopie übergeben, über Kopie wird ein neues Array angelegt, Kopie stirbt beim Verlassen der Methode, Original lebt unverändert weiter

```
public static char[] erstellenUndFuellen(String s) {
   char[] ret = new char[10];
   for (int i = 0; i < s.length(); i = i + 1)
      ret[i] = s.charAt(i);
   return ret;
}
```

In Methode wird neues Array angelegt, gefüllt und Zeiger auf dieses Array wird von Methode zurück gegeben

Geschachtelte oder mehrdimensionale Arrays

```
int[][] b = null;
b = new int[2][3];
b[0][0] = 1; b[1][2] = 1;
```




```
int[][] b = {{0, 1, 2}, {3, 4, 5}};
int summe = 0
for (int i = 0; i < b.length; i = i + 1)
   for (int j = 0; j < b[i].length; j = j + 1)
      summe = summe + b[i][j];
```

oder

```
for(int[] x: b)
   for(int y: x)
      summe = summe + y;
```

- Auch drei- und mehrdimensionale Arrays sind möglich
- Auch so etwas ist möglich (*nicht-rechteckiges Array*):

```
int[][] r = {{1}, {2, 3},{4, 5, 8}, {3, 2, 6, 1}};
```