



ARRAYS

Arrays

- Arrays: rudimentärste Art, mehrere gleichartige Objekte in Java zu speichern
 - *Elemente werden sequenziell hintereinander in den Hauptspeicher geschrieben*
 - *Zugriff auf ein Element durch Angabe der Index*
 - *Index eines Elementes: Position innerhalb des für den Array reservierten Speicherbereichs*
 - *Index beginnt stets mit 0*
 - *Index des letzten Element eines Arrays mit n Elementen ist stets $[n-1]$*

Arrays

■ Deklaration eines Arrays:

- *Datentyp gefolgt von einer geöffneten und einer geschlossenen eckigen Klammer und dem Bezeichner*

```
private Kunde[] kunden;  
...
```

Datentyp der Elemente

Name des Arrays

- *Bei der Deklaration wird die Größe des Arrays nicht angegeben. Es wird daher zu diesem Zeitpunkt noch kein Speicherplatz für den Array reserviert*

Arrays

- Bei der Instanziierung des Arrays wird Speicherplatz reserviert
 - *Instanziierung erfolgt mit dem Schlüsselwort new*
 - *In den eckigen Klammern ist die gewünschte Kapazität anzugeben*
 - *Bei der Instanziierung ist zu beachten, dass die Elemente mit dem Standardwert des jeweiligen Datentyps vorbelegt werden:*
 - Ein int-Array wird mit lauter Nullen gefüllt
 - Ein boolean-Array mit false-Werten
 - Arrays für komplexe Datentypen (z. B. Strings und eigene Klassen) mit null-Werten

Arrays

- Deklaration und Instanziierung eines Arrays

```
public class Kundenverwaltung {
```

```
    private Kunde[] kunden;
```

← Deklaration des Arrays

```
    ...
```

```
    public Kundenverwaltung(){
```

```
        kunden = new Kunde[42];
```

← Instanziierung mit Kapazität 42

Alle Elemente werden mit Datentyp-spezifischen Standard-Werten belegt

```
        System.out.println(kunden[0]);
```

```
        System.out.println(kunden[41]);
```

```
        System.out.println(kunden[42]);
```

← null (Standard-Wert)

← `ArrayIndexOutOfBoundsException`

```
}
```

Arrays

- Für eine andere Vorbelegung (keine Standardwerte) kann die Instanziierung auch mit einer Initialisierung einhergehen
 - *In geschweiften Klammern wird eine Komma-getrennte Liste von Wertaussprägungen angegeben*
 - *Durch Angabe der Initialwerte wird implizit die Kapazität des Arrays festgelegt - die Angabe der Kapazität fällt weg.*
 - *Die Initialisierung kann nur zusammen mit der Instanziierung erfolgen und nicht getrennt in einer späteren Anweisung*

Arrays

- Instanziierung eines Arrays mit Initialisierung

```
public class Kundenverwaltung {  
    private Kunde[] kunden;  
    ...  
    public Kundenverwaltung(){  
        kunden = new Kunde[] {new Kunde("Ulf", "Koll"),  
                                new Kunde("Ilse", "Stahl"),  
                                new Kunde("Rita", "Kafka")};  
        System.out.println(kunden[0]);  
        System.out.println(kunden[1]);  
        System.out.println(kunden[2]);  
    }  
}
```

Instanziierung mit Initialisierung

Kapazität wird implizit durch die Initialisierung vorgegeben

Rechteckiges Ausschneiden

Arrays

- Nach der Instanziierung kann die Kapazität eines Arrays nicht mehr verändert werden
- Überblick über die Kapazität mit Hilfe des Attribut `length` möglich
- Wichtig wenn eine separate Methode, in der die Größe des Arrays üblicherweise unbekannt ist, alle Elemente des Arrays verarbeiten möchte

Arrays

- Attribut length am Beispiel der for-Schleife

```
public class Kundenverwaltung {
```

```
    private Kunde[] kunden;
```

```
    ...
```

```
    public void aktualisiereAlleKunden(){
```

```
        for (int index=0; i<kunden.length; index++)
```

```
            if (kunden[index] != null)
```

```
                kundenSpeicher.aktualisieren(kunden[index]);
```

```
        }
```

```
    }
```

Schleife stoppt, sobald i kein gültiger Index mehr ist, d. h. $i == kunden.length$

Achtung: Prüfen, ob sich an der Index-Stelle tatsächlich ein Element befindet

pro Schleifendurchlauf wird ein Kunde aktualisiert

Arrays

- In Java ist es möglich, Arrays zu verschachteln:
 - *Die Elemente eines Arrays sind dann ebenfalls Arrays*
 - *Man spricht dann von mehrdimensionalen Arrays, da sich die Größe des Arrays bildlich gesehen nicht nur in eine Dimension ausdehnt, sondern in mindestens zwei*
 - *Möglicher Anwendungsfall: ein Schachbrett-Array, das zu jeder Zeile jeweils ein Array mit den dazugehörigen Spielfeldern enthält*

Arrays

- Ein und mehrdimensionale Arrays

```
int[] vektor = new int[] {2, 4, 1};  
  
int[][] matrix = new int[][] {{7, 3, 2},  
                               {9, 2, 7},  
                               {0, 3, 3},  
                               {1, 0, 6}};
```

2 Klammer-Paare
=> 2 Dimensionen

verschachtelter, 2-dimensionaler
Array

Arrays

■ Vorteile:

- *Deklaration und Verwendung unmittelbarer Bestandteil der Java-Syntax*
- *Daher nicht nötig Bibliotheken zu importieren*
- *Arrays können beliebige Typen enthalten: primitiven Datentypen, Strings und auch selbst programmierte Klassen*

Arrays

■ Nachteile:

- *Bei Arrays muss man sich selbst um die Kapazität kümmern -im Gegensatz zu Collections*
- *Array voll:*
 - *Es muss es zur Laufzeit mit einer größeren Kapazität neu initialisiert werden und alle Elemente müssen übertragen werden*
- *zu hohe Kapazität und folglich unnötigerweise ein viel zu großer Speicherbereich ebenfalls möglich*

Arrays

- Nachteile:
 - *Lücken in sortierten Arrays zu schließen ist mit großen Anstrengungen verbunden*
 - fürs Aufrücken muss jedes Folgeelement bewegt werden
 - *Arrays haben eine begrenzte eingebaute Funktionalität:*
 - zusätzlicher Programmieraufwand für die Form eines Stapels, einer Warteschlange oder einer Menge