

Java:

Arrays

Beispiel: Verzinstes Guthaben - Sparplan

Guthaben: 100 Euro Zinssatz: 10% p.a.

Jahr	Guthaben
0	100.00
1	110.00
2	121.00
3	133.10

Jahr	Guthaben
jahr[0]	100.00
jahr[1]	110.00
jahr[2]	121.00
jahr[3]	133.10

Beispiel: Verzinstes Guthaben - Sparplan

Guthaben: 100 Euro Zinssatz: 10% p.a.

Jahr	Guthaben
0	100.00
1	110.00
2	121.00
3	133.10

Jahr	Guthaben
jahr[0]	100.00
jahr[1]	110.00
jahr[2]	121.00
jahr[3]	133.10

Speichern in Variablen:

Möglichkeit 1: Für jedes Jahr eine Variable (jahr1, jahr2 ...)

Was spricht gegen diese Möglichkeit?

Array

Guthaben: 100 Euro Zinssatz: 10% p.a.

Jahr	Guthaben
0	100.00
1	110.00
2	121.00
3	133.10

Jahr	Guthaben
jahr[0]	100.00
jahr[1]	110.00
jahr[2]	121.00
jahr[3]	133.10

Speichern in Variablen:

Möglichkeit 1: Für jedes Jahr eine Variable (jahr1, jahr2 ...)

Möglichkeit 2: Benutzung einer Feldvariablen jahr[n]

Array
- 3 Elemente vom Typ double

Index	Wert
0	100.00
1	110.00
2	121.00

Array - Eigenschaften

Datentyp "**Feld**" oder "**Array**"

- erstes Element besitzt Index 0
- Größe wird bei Erzeugung angegeben, nachträglich nicht veränderbar!

Jahr	Guthaben
jahr[0]	100.00
jahr[1]	110.00
jahr[2]	121.00
jahr[3]	133.10

Array - Syntax

```
double[] guthaben = new double[3];  
// Definition + Erzeugung Array  
  
guthaben[0] = 100.00;  
guthaben[1] = 110.00;  
guthaben[2] = 121.00;  
// Wertezuweisung Array (nur im Konstruktor!)  
// 3. Element hat Index 2!
```

oder

```
double[] guthaben = {100.00, 110.00, 121.00};  
// Definition, Erzeugung, Initialisierung mit  
Werten  
// Größe wird durch Anzahl der Werte festgelegt
```

Aufgabe: Array definieren

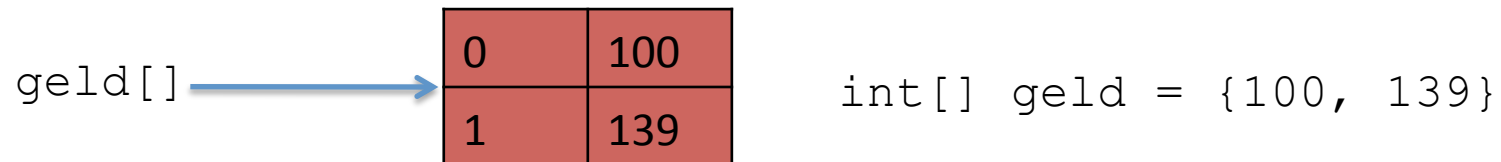
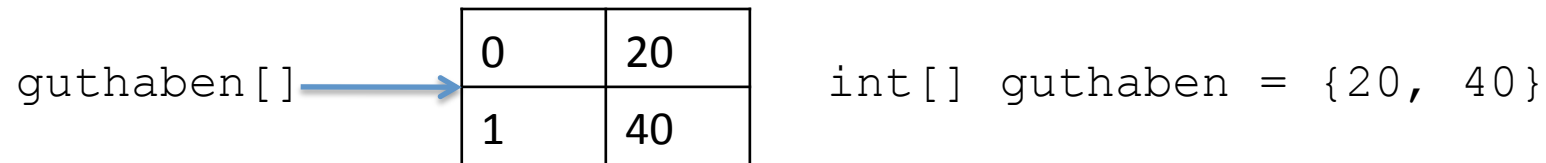
Sie kennen einen Börsentrick, mit dem Sie den Wert Ihres Aktiendepots jeden Monat verdoppeln. Uns interessiert immer nur der Erste des Monats.

Sie starten am 01.01.2014 mit 20 Euro. Am 01.04. zählen Sie Ihr Geld.

- 1) Wie viele Elemente enthält das Array?
- 2) Welchen Index hat das Element, das den 01.01.2014 repräsentiert?
- 3) Welchen Index hat das Element, das den 01.04.2014 repräsentiert?
- 4) Programmieren Sie ein int-Array mit den Werten zum jeweiligen Monatsanfang.
- 5) Geben Sie unter Verwendung der Arrayvariablen den Satz aus: "Zu Beginn hatte ich 20 Euro, am 01.04. habe ich 160 Euro."
- 6) Sie möchten alle Werte in der Konsole untereinander ausgeben. Verwenden Sie eine for-Schleife.
- 7) Befüllen Sie das Array mittels einer For-Schleife. Schreiben Sie dazu eine Methode `listebefuellen()`;
- 8) Übergeben Sie das Startkapital dem Konstruktor.

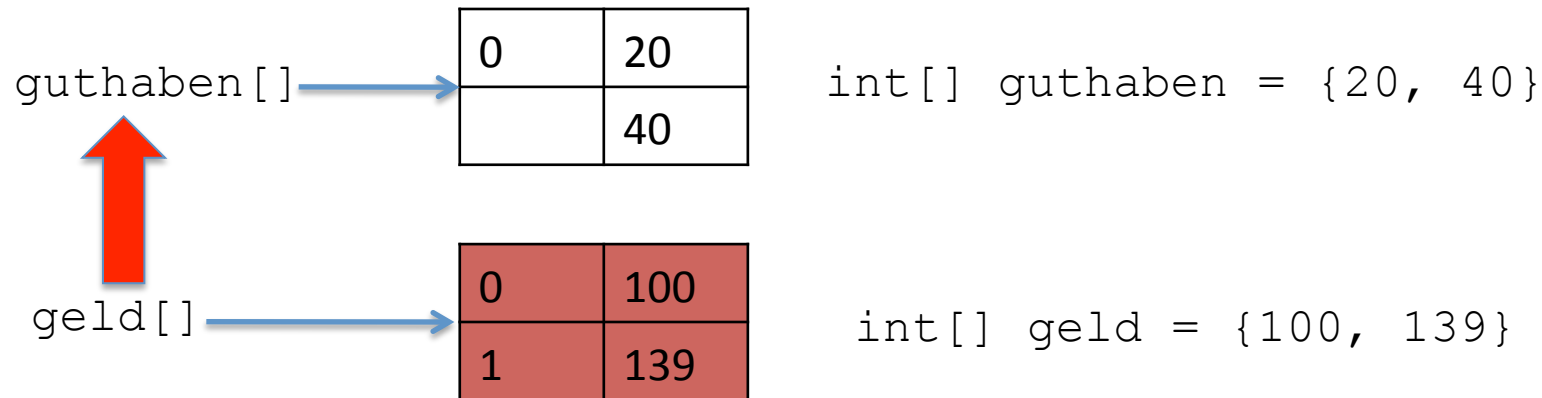
Arrays als Referenztypen

Arrays sind Referenztypen! D.h. sie verweisen lediglich auf Daten.



Arrays als Referenztypen

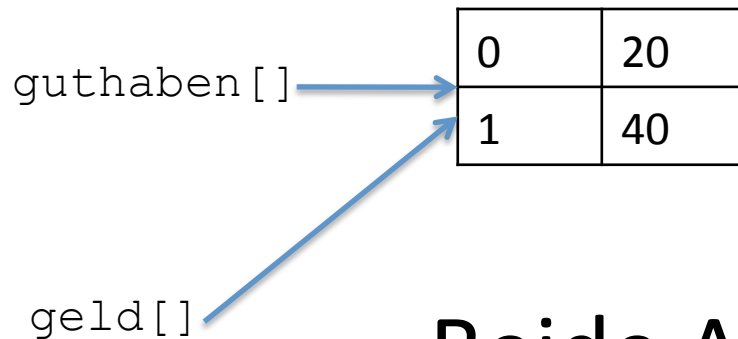
Arrays sind Referenztypen! D.h. sie verweisen lediglich auf Daten.



geld = guthaben;

Arrays als Referenztypen

Arrays sind Referenztypen! D.h. sie verweisen lediglich auf Daten.



**Beide Arrayvariablen
verweisen auf dasselbe
Array!**

→ Änderung in der einen Variablen
betreffen auch die andere
`geld[1] = 100` führt dazu, dass auch
`guthaben[1] == 100` ist!

Aufgabe: Array duplizieren

1. Erstellen Sie ein int-Array `primzahlen[]`, das die ersten 5 Primzahlen enthält.
2. Definieren Sie ein neues Array `primzahlenDuplikat[]` und erstellen Sie dort eine "Kopie" des Arrays `primzahlen[]` (`int[] primzahlenDuplikat = primzahlen`).
3. Verändern Sie in `primzahlen[]` Werte und überprüfen Sie, ob diese sich in `primzahlenDuplikat[]` ebenfalls geändert haben.
4. Sie können ein Array bspw. "richtig" kopieren, indem Sie auf das Array die Methode `clone()` anwenden. Probieren Sie das aus und überprüfen Sie, ob nun tatsächlich zwei verschiedene Arrays existieren.

Array mit Schleifen durchlaufen

Wichtig: Länge des Arrays kennen

```
int[] wetterdaten = {1,4,2,6}  
int anz = wetterdaten.length;  
  
System.out.println("Anzahl der Elemente: " + anz);
```

Array mit Schleifen durchlaufen

1) for-Schleife

```
// Beispiel 1

String[] name = {"Heinrich", "Margarete", "Joe"};

for(int i = 0; i < name.length; i++)
{
    System.out.println("Name " + i + ": " + name[i]);
}
```

Array mit Schleifen durchlaufen

1) for-Schleife

```
// Beispiel 2
```

```
int[] quadrat = new int[20];
```

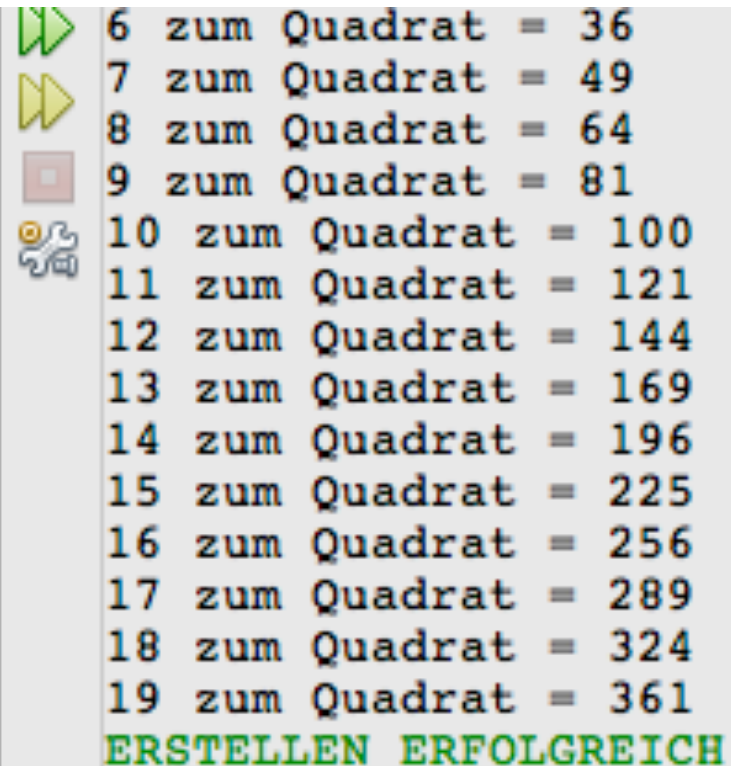
```
for(int i = 0; i < quadrat.length; i++)
```

```
{
```

```
    quadrat[i] = i * i;
```

```
    System.out.println(i+" zum Quadrat=" + quadrat[i]);
```

```
}
```

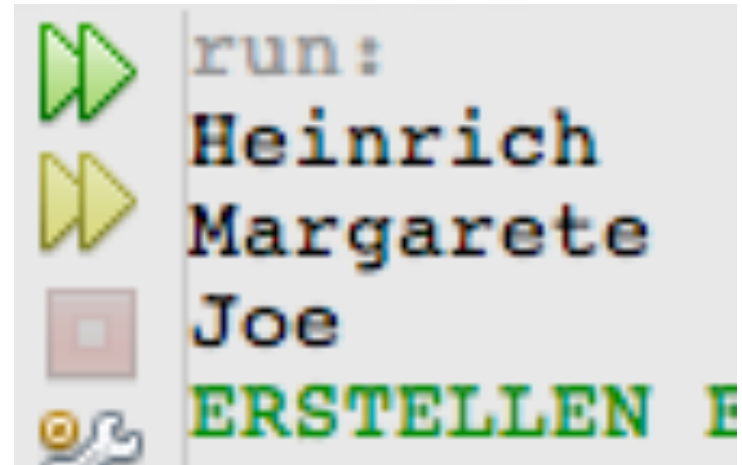


```
6 zum Quadrat = 36  
7 zum Quadrat = 49  
8 zum Quadrat = 64  
9 zum Quadrat = 81  
10 zum Quadrat = 100  
11 zum Quadrat = 121  
12 zum Quadrat = 144  
13 zum Quadrat = 169  
14 zum Quadrat = 196  
15 zum Quadrat = 225  
16 zum Quadrat = 256  
17 zum Quadrat = 289  
18 zum Quadrat = 324  
19 zum Quadrat = 361  
ERSTELLEN ERFOLGREICH
```

Array mit Schleifen durchlaufen

2) foreach-Schleife

nur AUSGABE von Werten EINES Arrays



```
String[] name = {"Heinrich", "Margarete", "Joe"};

for(String n : name)
{
    System.out.println(n);
}
```

for (String n : name)


durchläuft das Array; die Werte von **name** werden nacheinander in der neuen Variable **n** gespeichert.

Aufgabe: Array mit Schleifen durchlaufen

1. **Attribut:** Definieren Sie ein int-Array namens "tage" mit 365 Elementen.
2. **Schreiben Sie eine Methode „arrayBefuellen“.** Dort wird mittels einer Schleife das Array mit Werten befüllt. Der Wert der Array-Elemente soll jeweils dem Index + 1 entsprechen, also
tage[0] = 1
tage[1] = 2
...
tage[364] = 365

Aufgabe: Array mit Schleifen durchlaufen

1. **Attribut:** Definieren Sie ein int-Array namens "tage" mit 365 Elementen.
2. **Schreiben Sie eine Methode „arrayBefuellen“.** Dort wird mittels einer Schleife das Array mit Werten befüllt. Der Wert der Array-Elemente soll jeweils dem Index + 1 entsprechen, also
tage[0] = 1
tage[1] = 2
...
tage[364] = 365
3. **Schreiben Sie eine Methode „ausgeben“.** Lassen Sie den Inhalt des Arrays mittels einer foreach-Schleife ausgeben. Sie müssten die Werte 1 bis 365 erhalten.
4. Schreiben Sie eine Methode „listeAusgeben“. Dort wird eine Liste in folgendem Format ausgegeben:



```
run:
1 Tage haben 24 Stunden
2 Tage haben 48 Stunden
3 Tage haben 72 Stunden
4 Tage haben 96 Stunden
5 Tage haben 120 Stunden
6 Tage haben 144 Stunden
7 Tage haben 168 Stunden
8 Tage haben 192 Stunden
```

Mehrdimensionale Arrays

Jedes Array-Element enthält ein Array.

Mehrdimensionale Arrays



Sven Herzberg: Matroschka - CC BY-NC-SA 2.0

Matrjoschka
= Array mit 2 Elementen

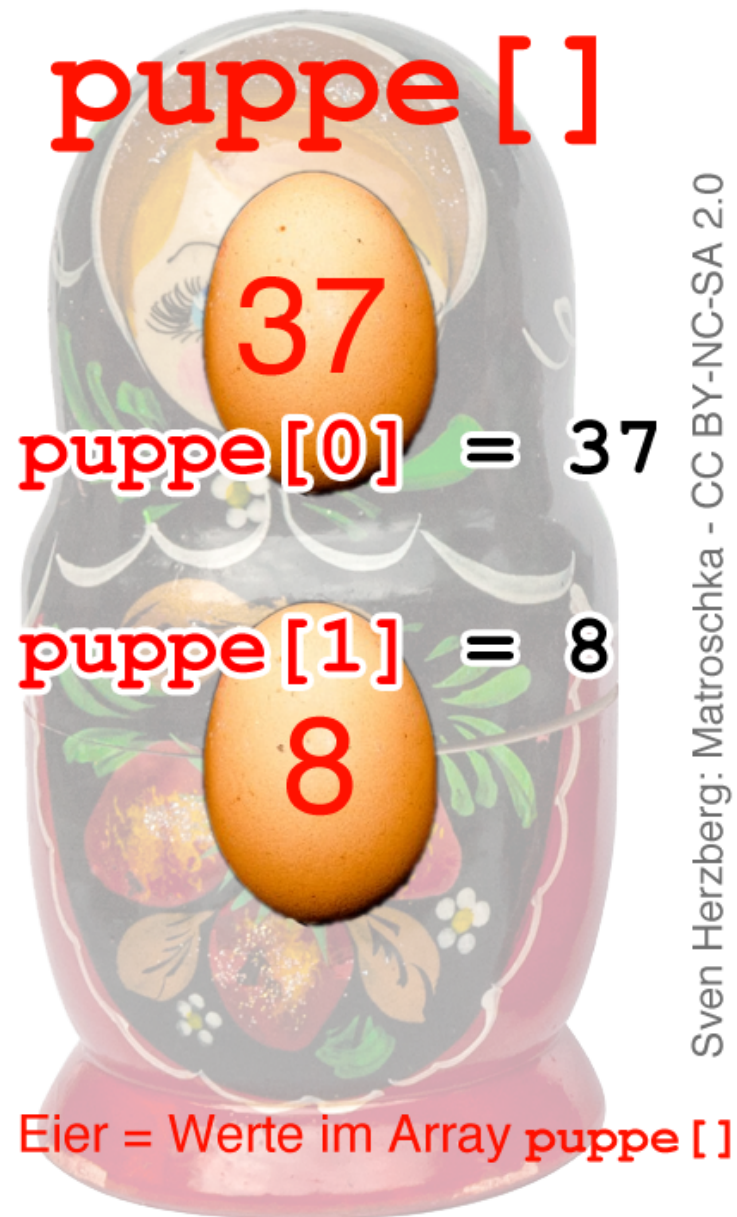
Mehrdimensionale Arrays



Sven Herzberg: Matroschka - CC BY-NC-SA 2.0

Matrjoschka
= Array mit 2 Elementen

Mehrdimensionale Arrays



eindimensionales („normales“) Array

Mehrdimensionale Arrays

2 int-Arrays, die nichts mit einander zu tun haben.



Sven Herzberg: Matroschka - CC BY-NC-SA 2.0

Mehrdimensionale Arrays



Das sind zwei einfache Arrays, die nichts miteinander zu tun haben.

Mehrdimensionale Arrays



Diese 2 Arrays stecken wir nun in unsere Matryoshka.

(Erinnerung:
Vorher waren 2 Eier drin,
Jetzt sind 2 Arrays drin.)

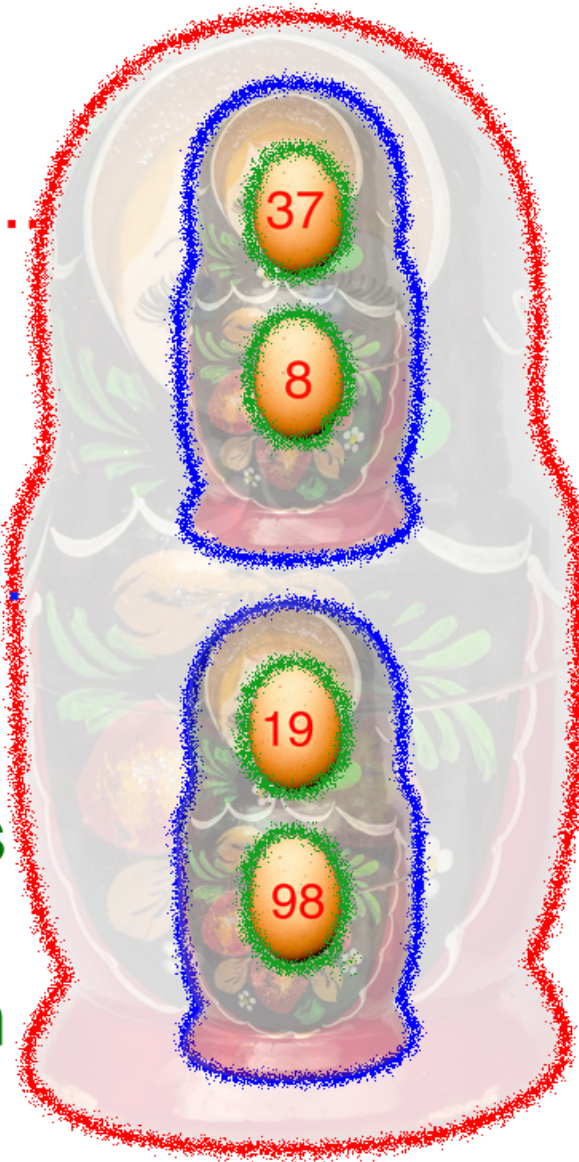
Sven Herzberg: Matroschka - CC BY-NC-SA 2.0

Mehrdimensionale Arrays

ein Array ...

beinhaltet
2 Arrays ...

die jeweils
2 Eier
beinhalten



Sven Herzberg: Matroschka - CC BY-NC-SA 2.0

Diese 2 Arrays stecken wir
nun in unsere Matroschka.

(Erinnerung:
Vorher waren 2 Eier drin,
Jetzt sind 2 Arrays drin.)

Mehrdimensionale Arrays

puppe



gerda



gerda[0]=37

gerda[1]=8

hanna



hanna[0]=19

hanna[1]=98

Namen:

Große Puppe: „puppe“

Kleine Puppe 1: „gerda“

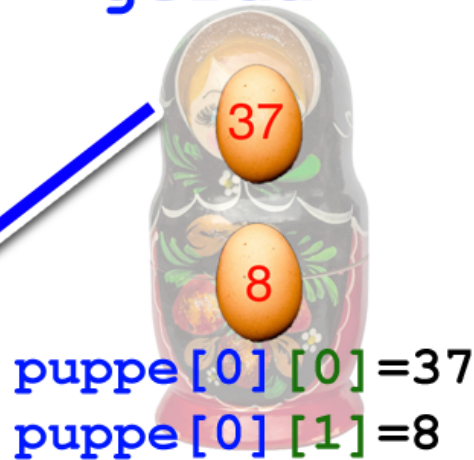
Kleine Puppe 2: „hanna“

Sven Herzberg: Matroschka - CC BY-NC-SA 2.0

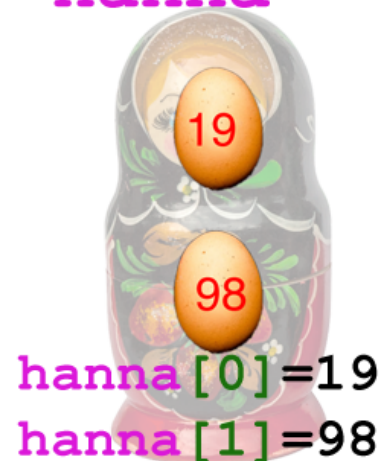
Mehrdimensionale Arrays

puppe

puppe[0]
gerda



hanna



gerda kommt in puppe

und wird damit zu einem
Element des Arrays „puppe“

Deshalb:
Umbenennung
„gerda“ → „puppe[0]“

Sven Herzberg: Matroschka - CC BY-NC-SA 2.0

Mehrdimensionale Arrays

puppe

puppe[0]
gerda



puppe[0][0]=37
puppe[0][1]=8

puppe[1]
hanna

puppe[1][0]=19
puppe[1][1]=98

Sven Herzberg: Matroschka - CC BY-NC-SA 2.0

gerda kommt in puppe

und wird damit zu einem
Element des Arrays „puppe“

Deshalb:

Umbenennung

„gerda“ → „puppe[0]“

hanna kommt in puppe

und wird damit zu einem
Element des Arrays „puppe“

Deshalb:

Umbenennung

„hanna“ → „puppe[1]“

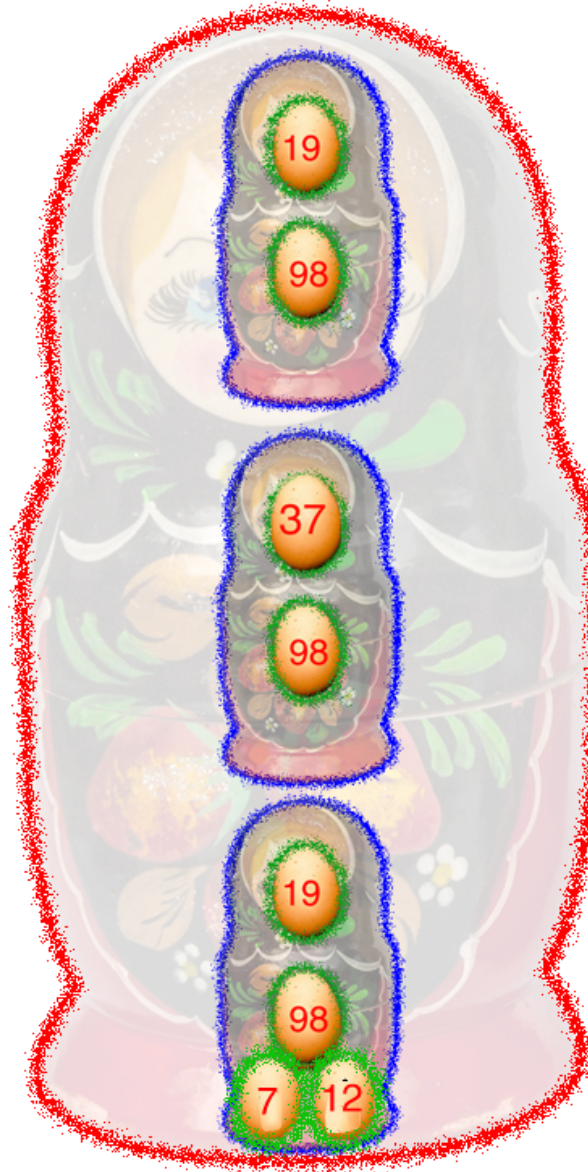
Mehrdimensionale Arrays

Größe der Arrays kann variieren! (Datentyp nicht!)

```
puppe[0][0] = 19  
puppe[0][1] = 98
```

```
puppe[1][0] = 37  
puppe[1][1] = 98
```

```
puppe[2][0] = 19  
puppe[2][1] = 8  
puppe[2][2] = 7  
puppe[2][3] = 12
```



Sven Herzberg: Matroschka - CC BY-NC-SA 2.0

Mehrdimensionale Arrays

Programmierung:

```
String[] einfach = new String[2]
```

0	"wert 1"
1	"wert 2"

einfach[0] → "wert 1"

einfach[1] → "wert 2"

```
String[][] mehrfach = new String[2][3]
```

0	0	"wert a"
	1	"wert b"
	2	"wert c"
1	0	"wert d"
	1	"wert e"
	2	„wert f“

mehrfach[0][0] → "wert a"

mehrfach[0][1] → "wert b"

mehrfach[0][2] → "wert c"

mehrfach[1][0] → "wert d"

mehrfach[1][1] → "wert e"

mehrfach[1][2] → "wert f"

Mehrdimensionale Arrays – Einfache Aufgabe

Wir wollen die Personen Gerda Schmitt, Hanna Müller und Martha Metz speichern.

- 1) Programmieren Sie ein eindimensionales („normales“) String-Array namens „puppe“. Dieses Array enthält die Werte „gerda“, „hanna“, „martha“
- 2) Erweitern Sie das Array um eine Dimension. gerda, hanna und martha sollen jeweils ein Array sein, das zwei Werte enthält, nämlich den Vor- und Nachnamen.
- 3) Testen Sie die Lauffähigkeit.
`System.out.println(puppe[0][0]);`
sollte „Gerda“ ausgeben,
`System.out.println(puppe[0][1]);`
sollte „Schmitt“ ausgeben.
- 4) Schreiben Sie eine Schleife, die alle Namen untereinander ausgibt:

Gerda Schmitt
Hanna Müller
Martha Metz

Mehrdimensionale Arrays: Beispiel "Tabelle"

Erste Dimension = Zeile

Zweite Dimension = Spalte

	0	1	2
0	zeile 0, spalte 0	zeile 0, spalte 1	zeile 0, spalte 2
1	zeile 1, spalte 0	zeile 1, spalte 1	zeile 1, spalte 2

```
// Dimensionen definieren
```

```
int anzahlZeilen = 2;  
int anzahlSpalten = 3;
```

```
String[][] tabelle = new String[anzahlZeilen][anzahlSpalten];
```


Mehrdimensionale Arrays: Beispiel "Tabelle"

Erste Dimension = Zeile

Zweite Dimension = Spalte

	0	1	2
0	zeile 0, spalte 0	zeile 0, spalte 1	zeile 0, spalte 2
1	zeile 1, spalte 0	zeile 1, spalte 1	zeile 1, spalte 2

```
// Dimensionen definieren
```

```
int anzahlZeilen = 2;  
int anzahlSpalten = 3;
```

```
String[][] tabelle = new String[anzahlZeilen][anzahlSpalten];
```

```
System.out.println("Array-Länge: " + tabelle.length);  
// Ausgabe: 2 (denn in der ersten Dimension 2 Elemente)
```

```
System.out.println("Länge der Dimension 0: "+tabelle[0].length);  
// Ausgabe: 3 (denn erstes Element enthält 3 Elemente)
```

Mehrdimensionale Arrays: Beispiel "Tabelle"

Erste Dimension = Zeile

Zweite Dimension = Spalte

	0	1	2
0	zeile 0, spalte 0	zeile 0, spalte 1	zeile 0, spalte 2
1	zeile 1, spalte 0	zeile 1, spalte 1	zeile 1, spalte 2

```
int anzahlZeilen = 2;
int anzahlSpalten = 3;
String[][] tabelle = new String[anzahlZeilen][anzahlSpalten];

// Tabelle füllen

// Zeilen durchlaufen (= 1. Dimension)
for(int zeile = 0; zeile < anzahlZeilen; zeile++) {
    // Spalten durchlaufen (= 2. Dimension)
    for(int spalte = 0; spalte < anzahlSpalten; spalte++) {
        tabelle[zeile][spalte] = "zeile " + zeile + ", spalte " +
        spalte";
    }
}
```

Mehrdimensionale Arrays: Beispiel "Tabelle"

Erste Dimension = Zeile

Zweite Dimension = Spalte

	0	1	2
0	zeile 0, spalte 0	zeile 0, spalte 1	zeile 0, spalte 2
1	zeile 1, spalte 0	zeile 1, spalte 1	zeile 1, spalte 2

```
int anzahlZeilen = 2;
int anzahlSpalten = 3;
String[][] tabelle = new String[anzahlZeilen][anzahlSpalten];
// Tabelle füllen

// Zeilen durchlaufen (= 1. Dimension)
for(int zeile = 0; zeile < anzahlZeilen; zeile++) {
    // Spalten durchlaufen (= 2. Dimension)
    for(int spalte = 0; spalte < anzahlSpalten; spalte++) { tabelle[zeile][spalte] = "zeile " + zeile + ", spalte " + spalte; }
}
// Werte ausgeben

// Zeilen durchlaufen (= 1. Dimension)
for(int zeile = 0; zeile < anzahlZeilen; zeile++) {
    // Spalten durchlaufen (= 2. Dimension)
    for(int spalte = 0; spalte < anzahlSpalten; spalte++) {
        System.out.printf("Wert für Zeile %d, Spalte %d: %s\n", zeile,
        spalte, tabelle[zeile][spalte]);
    }
}
```

Aufgabe: Mehrdimensionales Array

1. Erstellen Sie ein zweidimensionales String-Array `"januarKalender"`. Die erste Dimension enthält die 31 Tage, die zweite Dimension die Stunden jeden Tages (nämlich 24 pro Tag).
2. Der Wert für alle Stunden ist gleich, nämlich "nichts". Schreiben Sie eine Methode `kalenderMitNichtswertBefuellen()`, die allen Elementen im Array den Wert „nichts“ zuweist. (Sie müssen keine Ausgabe vornehmen!)
3. Schreiben Sie eine Methode `alleWerteAusgeben()`, die alle Werte im Array ausgibt.
4. Tragen Sie Ihre Termine ein:
 1. Januar, 5. Stunde: "Aufstehen!!!"
`januarKalender[0][4] = "Aufstehen!!!!";`
 1. Januar, 10. Stunde: "Mittagessen"
 31. Januar, 23. Stunde: "Gute Nacht!"
5. Lassen Sie sich mithilfe einer Schleife die 24 Stunden des 1. Januar komplett ausgeben.
 1. Januar, 0 Uhr: nichts
 2. Januar, 1 Uhr: nichts
 - ...
6. Lassen Sie sich mithilfe einer Schleife alle Tage des Januars wie in der vorigen Aufgabe ausgeben.
7. Das FBI will Ihren Kalender durchforsten! Löschen Sie alle Termine.

Objekt-Arrays

Arrays können als Werte natürlich auch Objekte beherbergen (wie z.B. bei String-Array der Fall):

Array enthält String-Werte

Index	Wert	Syntax
0	"Fiffi"	<code>hund[0] = "Fiffi";</code>
1	"Rex"	<code>hund[1] = "Rex";</code>
2	"Hasso"	<code>hund[2] = "Hasso";</code>

Objekt-Arrays

Arrays können als Werte natürlich auch Objekte beherbergen (wie z.B. bei String-Array der Fall):

Array enthält String-Werte

Index	Wert	Syntax
0	"Fiffi"	<code>hund[0] = "Fiffi";</code>
1	"Rex"	<code>hund[1] = "Rex";</code>
2	"Hasso"	<code>hund[2] = "Hasso";</code>

Array enthält Objekte vom Typ "Hund" (`hund1 = new Hund();`)

Index	Wert	Syntax
0	hund1	<code>hund[0] = hund1;</code>
1	hund2	<code>hund[1] = hund2;</code>
2	hund3	<code>hund[2] = hund3;</code>

Objekt-Arrays

```
public class Hund
{
    public String name;
    public int alter;

    public Hund(String name, int alter){
        this.name      = name;
        this.alter      = alter;
    }

    public void huepf(){
        if(this.alter < 20){
            System.out.println(this.name + " - HÜPF! bin erst " +
                               this.alter); }

        else{
            System.out.println(this.name + " - ächz! bin schon " +
                               this.alter);
        }
    }
}
```

Objekt-Arrays

Arrays können als Werte natürlich auch Objekte beherbergen (wie z.B. bei String-Array der Fall):

```
public class Hund
{
    public String name;
    public int alter;
    public Hund(String name, int
alter){
        this.name = name;
        this.alter = alter;
    }
    public void huepf(){
        if(this.alter < 20){

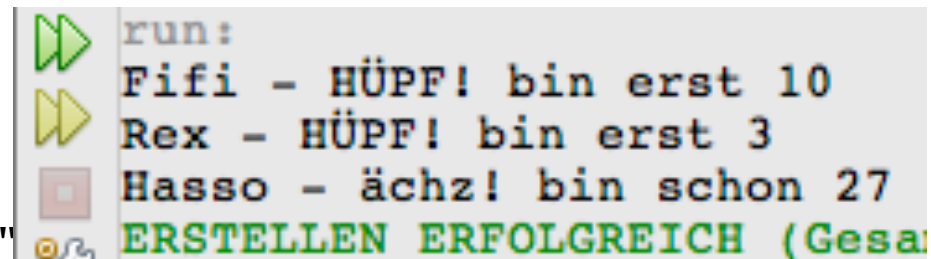
            System.out.println(this.name + " -
HÜPF! bin erst " + this.alter);
        }
        else{

            System.out.println(this.name + "
ächz! bin schon " + this.alter);
        }
    }
}
```

```
Hund a = new Hund("Fifi", 10);
Hund b = new Hund("Rex", 3);
Hund c = new Hund("Hasso", 27);
```

```
Hund[] hundeArray = {a,b,c};
```

```
hundeArray[0].huepf();
hundeArray[1].huepf();
hundeArray[2].huepf();
```



```
run:
Fifi - HÜPF! bin erst 10
Rex - HÜPF! bin erst 3
Hasso - ächz! bin schon 27
ERSTELLEN ERFOLGREICH (Gesamtzeit: 0:00:00)
```


Aufgabe 1: Array mit Objekten

Vollziehen das eben genannte Beispiel „Hund“ nach:

- Klasse „Hund“ schreiben (s.u.)
- Startklasse: 3 Objekte d. Klasse Hund erzeugen
- Startklasse: Array „hundeliste“ erzeugen
- Startklasse: Die 3 Hund-Objekte dem Array „hundeliste“ hinzufügen
- Methoden der Hund-Klasse auf die Elemente des Arrays anwenden

Hund
-name:String -alter:int
Hund(name:String, alter:int) +huepf():void +datenAusgeben():void // gibt name + alter aus

Aufgabe 2: Array mit Objekten

1. Programmieren Sie die Klasse Rechnung:

Rechnung
-rechnungsNummer:int -rechnungsBetrag:float
+Rechnung(rechnungsNummer:int, rechnungsBetrag:float) (getter, setter)

2. Erzeugen Sie 3 Objekte dieser Klasse (jeweils mit eigener Rechnungsnummer und eigenem Rechnungsbetrag).
3. Erstellen Sie ein Array "rechnungen", das diese 3 Objekte abspeichert.
4. Wenden Sie die Settermethoden auf ein Element an und verändern Sie die Rechnungsnummer und den Rechnungsbetrag. Lassen Sie sich beides über Gettermethoden ausgeben.

Objekt-Arrays mit Schleifen durchlaufen

„normales“ Array durchlaufen

```
String[] schuelerListe = new String[2];

schuelerListe[0] = "Dascha";
schuelerListe[1] = "Mascha";

for(int i = 0; i < schuelerListe.length; i++)
{
    System.out.println("Schüler/in " +

        i + " " +
        schuelerListe[i]);
}
```

Objekt-Arrays mit Schleifen durchlaufen

Array mit Objekten durchlaufen

```
Schueler s1 = new Schueler("Mascha", "Müller");  
Schueler s2 = new Schueler("Dascha",  
"Donnerberger");
```

```
Schueler[] schuelerListe = new Schueler[2];
```

```
schuelerListe[0] = s1;  
schuelerListe[1] = s2;
```

```
for(int i = 0; i < schuelerListe.length; i++) {  
    System.out.println("Schüler/in " + i + " " +  
        schuelerListe[i].getNachname());  
}
```

Objekt-Arrays mit Schleifen durchlaufen

```
String[] schuelerListe = new String[2];

schuelerListe[0] = "Dascha";
schuelerListe[1] = "Mascha";

for(int i = 0; i < schuelerListe.length; i++)    {
    System.out.println("Schüler/in " + i + " " +
schuelerListe[i]);                               }
```

Array mit Objekten durchlaufen:

```
Schueler s1 = new Schueler("Mascha", "Müller");
Schueler s2 = new Schueler("Dascha", "Donnerberger");
```

```
Schueler[] schuelerListe = new Schueler[2];

schuelerListe[0] = s1;
schuelerListe[1] = s2;

for(int i = 0; i < schuelerListe.length; i++)    {
    System.out.println("Schüler/in " + i + " " +
schuelerListe[i].getNachname());                }
```

Objekt-Arrays mit Schleifen durchlaufen

```
String[] schuelerListe = new String[2];
```

```
schuelerListe[0] = "Dascha";  
schuelerListe[1] = "Mascha";
```

```
for(int i = 0; i < schuelerListe.length; i++)  
    System.out.println("Schüler/in " + i + " " +  
schuelerListe[i]);
```

Array mit Objekten durchlaufen

```
Schueler s1 = new Schueler("Mascha", "Müller");  
Schueler s2 = new Schueler("Dascha", "Donnerberger");
```

```
Schueler[] schuelerListe = new Schueler[2];
```

```
schuelerListe[0] = s1;  
schuelerListe[1] = s2;
```

```
for(int i = 0; i < schuelerListe.length; i++)  
    System.out.println("Schüler/in " + i + " " +  
schuelerListe[i].getNachname());
```

Objekt-Arrays mit Schleifen durchlaufen

```
String[] schuelerListe = new String[2];
```

```
schuelerListe[0] = "Dascha";  
schuelerListe[1] = "Mascha";
```

```
for(int i = 0; i < schuelerListe.length; i++)  
    System.out.println("Schüler/in " + i + " " +  
schuelerListe[i]);
```

Array mit Objekten durchlaufen

```
Schueler s1 = new Schueler("Mascha", "Müller");  
Schueler s2 = new Schueler("Dascha", "Donnerberger");
```

```
Schueler[] schuelerListe = new Schueler[2];
```

```
schuelerListe[0] = s1;  
schuelerListe[1] = s2;
```

```
for(int i = 0; i < schuelerListe.length; i++)  
    System.out.println("Schüler/in " + i + " " +  
schuelerListe[i].getNachname());
```

Objekt-Arrays mit Schleifen durchlaufen

```
String[] schuelerListe = new String[2];
```

```
schuelerListe[0] = "Dascha";  
schuelerListe[1] = "Mascha";
```

```
for(int i = 0; i < schuelerListe.length; i++)  
    System.out.println("Schüler/in " + i + " " +  
schuelerListe[i]);
```

Array mit Objekten durchlaufen

```
Schueler s1 = new Schueler("Mascha", "Müller");  
Schueler s2 = new Schueler("Dascha", "Donnerberger");
```

```
Schueler[] schuelerListe = new Schueler[2];
```

```
schuelerListe[0] = s1;  
schuelerListe[1] = s2;
```

```
for(int i = 0; i < schuelerListe.length; i++)  
    System.out.println("Schüler/in " + i + " " +  
schuelerListe[i].getNachname());
```


Array mit Schleifen durchlaufen

(benutzen Sie das Array „rechnungen“ aus der letzten Aufgabe)

Geben Sie in einer Liste alle Rechnungsbeträge aus.
Geben Sie in einer Liste alle Rechnungsnummern aus.

Fortgeschrittene:

Geben Sie den Durchschnitt aller Rechnungsbeträge aus.

Array mit Schleifen durchlaufen 2

(siehe Java_30-Arrays-5c: Array mit Objekten – Abschlussübung)

- Klasse „Fussballer“ anlegen
- für jeden Fußballer ein Objekt erzeugen
- Fußballer-Objekte in Array speichern
- Fußballer-Daten ausgeben

Zusatz:

Durchschnittswerte ausrechnen,
bei positiver Abweichung ausgeben

