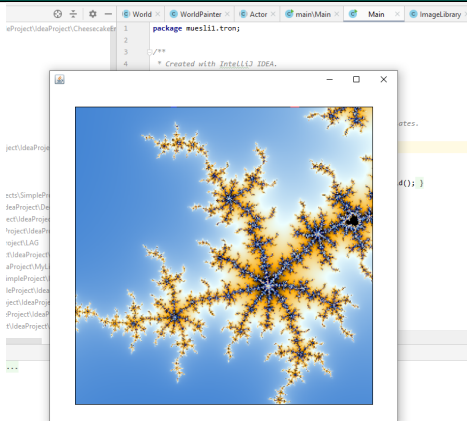


# FOP Recap #3



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

## Arrays





---

# Weiter gehts!



Organisatorisches

Wiederholung

Arrays – Basics

Eindimensionale Arrays

Mehrdimensionale Arrays

# Das steht heute auf dem Plan



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Organisatorisches  
Feedback

Wiederholung

Arrays – Basics

Eindimensionale Arrays

Mehrdimensionale Arrays



- Wir danken für die Rückmeldung
- Das Live-Coding wird diesmal anders aufgebaut sein
- Je nach Rückmeldung werden die Änderungen beibehalten/weitere Anpassungen vorgenommen
- Weiteres Feedback ist natürlich nach wie vor willkommen

# Das steht heute auf dem Plan



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Organisatorisches

Wiederholung

Objekte

Attribute: Standardwerte

Werte speichern

Arrays – Basics

Eindimensionale Arrays

Mehrdimensionale Arrays



### Definition – Objekt:

Ein **Objekt** ist eine Instanz einer **Klasse**. Es hat seinen eigenen Speicher und die gespeicherten Werte in seinen Attributen können unabhängig von anderen Instanzen geändert werden.

Auf Objekten können **Methoden** aufgerufen werden. Dies geht auf primitiven Datentypen nicht. Welche **Methoden** verfügbar sind, hängt von der **Klasse** ab.

Um ein neues Objekt zu erstellen, muss der **new**-Operator verwendet werden. Hierbei wird der jeweilige **Konstruktor** auf dem neu erstellten Objekt implizit aufgerufen.

Nicht mehr genutzte Objekte werden von Java (irgendwann) automatisch gelöscht [Stichwort Garbage-Collection]

```
1 LemonTree tree = new LemonTree(3);
```

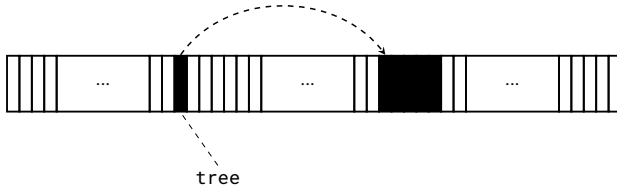


Abbildung: Abstrakte Visualisierung des Speichers





```
1 LemonTree tree = new LemonTree(3);
2 LemonTree tree2 = tree;
3 tree = new LemonTree(56);
4 System.out.println(tree.numberOfLemons);
5 System.out.println(tree2.numberOfLemons);
6 tree2 = tree;
7 System.out.println(tree.numberOfLemons);
```

```
$ 56
$ 3
$ 56
```

```
1 LemonTree tree = new LemonTree(3);  
2 LemonTree tree2 = tree;
```

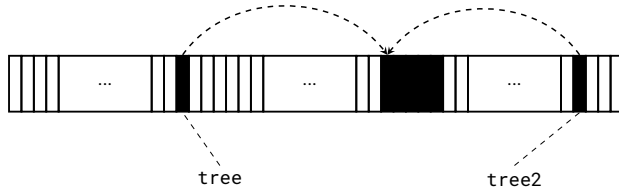


Abbildung: Abstrakte Visualisierung des Speichers

```
1 tree = new LemonTree(56);  
2 tree2 = tree;
```

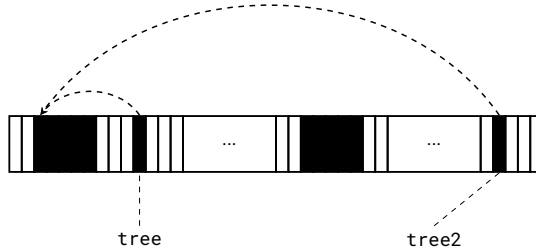


Abbildung: Abstrakte Visualisierung des Speichers

# Wiederholung

## Attribute: Standardwerte



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Name	Typ	Attribut Standardwert
boolean	Wahr/Falsch	false
int	Ganze Zahl	0
double	Gleitkommazahl	0.0
String	Zeichenkette	null
LemonTree	Eigene Klasse	null

# Wiederholung

## Werte speichern



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

- In einer lokale Variable
- Als Attribut in einem Objekt
- Neu: In einem Array

# Das steht heute auf dem Plan



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Organisatorisches

Wiederholung

Arrays – Basics

- Arrays mit Primitivem Komponententyp

- length

- Exceptions

- Arrays mit Referenztypen

Eindimensionale Arrays

Mehrdimensionale Arrays



## Definition – Array:

Ein **Array** hat einen **Komponententyp** und eine feste **Länge**, welche bei der Erstellung angegeben werden muss.

In ihm können **mehrere Werte** von dem angegebenen **Komponententypen** gespeichert werden.

Der Zugriff erfolgt über einen **Index**, der mit **0** anfängt und bei **length - 1** aufhört.

Eine Variable / ein Attribut mit dem Typ Array kann auch den Wert **null** annehmen.

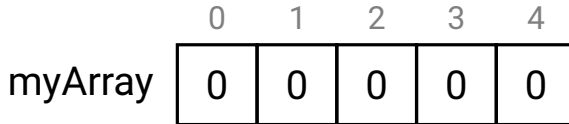
# Arrays – Basics

## Arrays mit Primitivem Komponententyp



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int[] myArray = new int[5];
```





# Arrays – Basics

## Arrays mit Primitivem Komponententyp



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1  int[] myArray = new int[5];  
2  myArray[0] = 1;  
3  myArray[1] = -5;  
4  myArray[2] = 9;  
5  myArray[3] = 3;  
6  myArray[4] = 5;
```

	0	1	2	3	4
myArray	1	-5	9	3	5

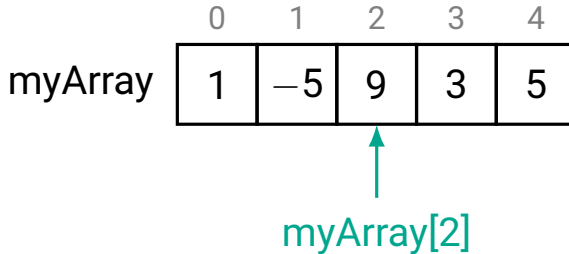
# Arrays – Basics

## Arrays mit Primitivem Komponententyp



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int[] myArray = new int[] {1, -5, 9, 3, 5};
```



# Arrays – Basics

length



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int[] myArray = new int[5];  
2 myArray[1] = 3;
```

```
1 System.out.println(myArray.length);  
2 System.out.println(myArray[0]);  
3 System.out.println(myArray[1]);
```

# Arrays – Basics

length



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int[] myArray = new int[5];  
2 myArray[1] = 3;
```

```
1 System.out.println(myArray.length);  
2 System.out.println(myArray[0]);  
3 System.out.println(myArray[1]);
```

```
$ 5  
$ 0  
$ 3
```

# Arrays – Basics

## Exceptions

```
1  int[] myArray = null;
```

```
1  System.out.println(myArray.length); // NullPointerException!  
2  System.out.println(myArray[0]); // NullPointerException!  
3  System.out.println(myArray[1]); // NullPointerException!
```

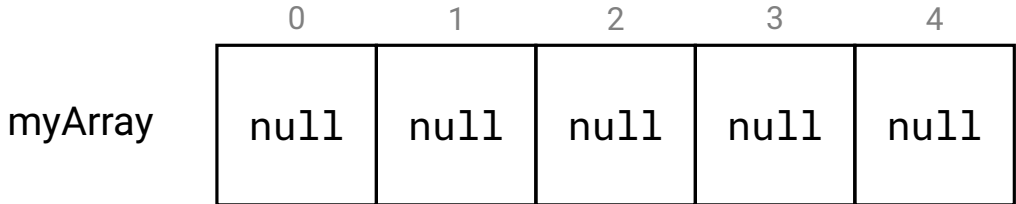
```
1  int[] myArray = new int[25];
```

```
1  // ArrayIndexOutOfBoundsException on all lines:  
2  System.out.println(myArray[-8]);  
3  System.out.println(myArray[-1]);  
4  System.out.println(myArray[25]);  
5  System.out.println(myArray[400]);
```

# Arrays – Basics

## Arrays mit Referenztypen

```
1 Robot[] myArray = new Robot[5];
```



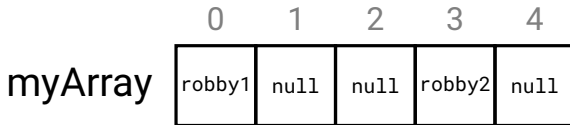
# Arrays – Basics

## Arrays mit Referenztypen



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 Robot robby1 = new Robot(0, 0, Direction.UP, 5);  
2 Robot robby2 = new Robot(3, 3, Direction.UP, 6);  
3 Robot[] myArray = new Robot[5];  
4 myArray[0] = robby1;  
5 myArray[3] = robby2;
```





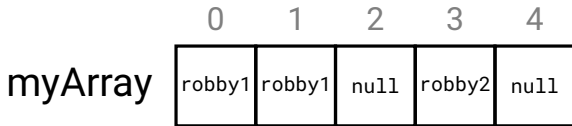
# Arrays – Basics

## Arrays mit Referenztypen



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 Robot robby1 = new Robot(0, 0, Direction.UP, 5);
2 Robot robby2 = new Robot(3, 3, Direction.UP, 6);
3 Robot[] myArray = new Robot[5];
4 myArray[0] = robby1;
5 myArray[1] = robby1;
6 myArray[3] = robby2;
7 System.out.println(myArray[0] == myArray[1]); // true
8 System.out.println(myArray[0] == myArray[3]); // false
```



# Das steht heute auf dem Plan



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Organisatorisches

Wiederholung

Arrays – Basics

Eindimensionale Arrays  
Über Elemente iterieren  
Enhanced for-Loops

Mehrdimensionale Arrays

# Eindimensionale Arrays

## Über Elemente iterieren



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int[] numArray = new int[250];
```

```
1 for(int i = 0; i < numArray.length; i++) {  
2     int value = numArray[i];  
3     System.out.println(value);  
4 }
```

# Eindimensionale Arrays

## Enhanced for-Loops



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int[] numArray = new int[250];
```

```
1 for(int value : numArray) {  
2     System.out.println(value);  
3 }
```

- Achtung: Kein (direkter) Zugriff auf Index möglich!
- Gut fürs Lesen, schlecht fürs Schreiben

# Das steht heute auf dem Plan



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Organisatorisches

Wiederholung

Arrays – Basics

Eindimensionale Arrays

**Mehrdimensionale Arrays**  
Über Elemente iterieren  
Enhanced for



```
1  int[][] multiArray = new int[3][5]; // length == 3
2  int[] a = multiArray[0]; // length == 5
3  int[] b = multiArray[1]; // length == 5
4  int[] c = multiArray[2]; // length == 5
```

- Zweidimensionale Arrays sind 'Arrays von Arrays'

# Mehrdimensionale Arrays

## Über Elemente iterieren



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1  int[][] multiArray = new int[3][5];
```

```
1  for(int i = 0; i < 3; i++) {  
2      int[] subArray = multiArray[i];  
3      for(int j = 0; j < 5; j++) {  
4          int value = subArray[j];  
5      }  
6  }
```

# Mehrdimensionale Arrays

## Über Elemente iterieren



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1  int[][] multiArray = new int[3][5];
```

```
1  for(int i = 0; i < 3; i++) {  
2      for(int j = 0; j < 5; j++) {  
3          int value = subArray[i][j];  
4      }  
5  }
```



# Mehrdimensionale Arrays

Enhanced for



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1  int[][] multiArray = new int[3][5];
```

```
1  for(int[] subArray : multiArray) {  
2      for(int value : subArray) {  
3          System.out.println(value);  
4      }  
5  }
```



---

# Live-Coding!