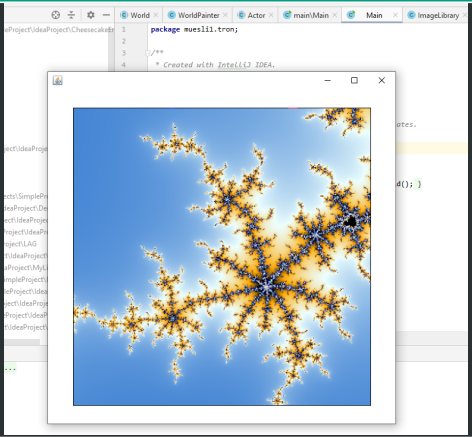
FOP Recap #3

Arrays



Weiter gehts!

Heute im Recap

TECHNISCHE UNIVERSITÄT DARMSTADT

Organisatorisches

Wiederholung

Arrays – Basics

Eindimmensionale Arrays

Mehrdimensionale Arrays

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

- Organisatorisches
  - Feedback
- Wiederholung
- Arrays – Basics
- Eindimmensionale Arrays
- Mehrdimensionale Arrays

Organisatorisches
Feedback

TECHNISCHE UNIVERSITÄT DARMSTADT

- 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

Eindimmensionale Arrays

Mehrdimensionale Arrays

Wiederholung
Objekte

TECHNISCHE UNIVERSITÄT DARMSTADT

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]

Wiederholung
Objekte

TECHNISCHE UNIVERSITÄT DARMSTADT

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

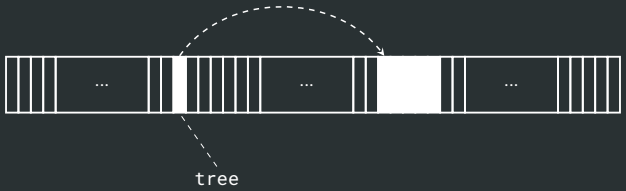


Abbildung: Abstrakte Visualisierung des Speichers

Wiederholung

Objekte

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
LemonTree tree = new LemonTree(3);
LemonTree tree2 = tree;
tree = new LemonTree(56);
System.out.println(tree.numberOfLemons);
System.out.println(tree2.numberOfLemons);
tree2 = tree;
System.out.println(tree.numberOfLemons);
```

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

Wiederholung
Objekte

TECHNISCHE UNIVERSITÄT DARMSTADT

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

Abbildung: Abstrakte Visualisierung des Speichers

Wiederholung
Objekte

TECHNISCHE UNIVERSITÄT DARMSTADT

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

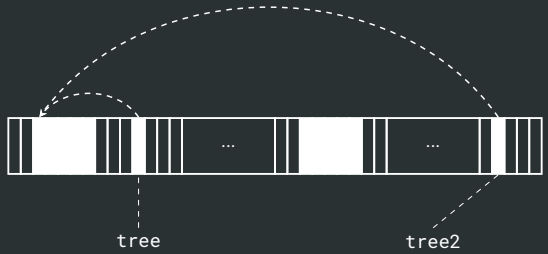


Abbildung: Abstrakte Visualisierung des Speichers

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

Eindimmensionale Arrays

Mehrdimensionale Arrays

Arrays – Basics

TECHNISCHE UNIVERSITÄT DARMSTADT

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.

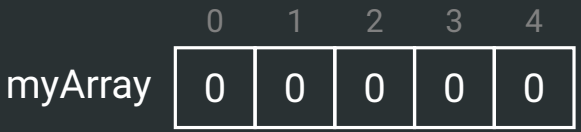
Eine Variable / ein Attribut mit dem Typ Array kann auch den Wert **`null`** annehmen.

Arrays – Basics

Arrays mit Primitivem Komponententyp

TECHNISCHE UNIVERSITÄT DARMSTADT

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



Arrays – Basics

Arrays mit Primitivem Komponententyp

TECHNISCHE UNIVERSITÄT DARMSTADT

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

Arrays – Basics

Arrays mit Primitivem Komponententyp

TECHNISCHE UNIVERSITÄT DARMSTADT

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

Arrays – Basics

length

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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

Arrays – Basics

length

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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

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

Arrays – Basics

Exceptions

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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

Arrays – Basics

Exceptions

TECHNISCHE UNIVERSITÄT DARMSTADT

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

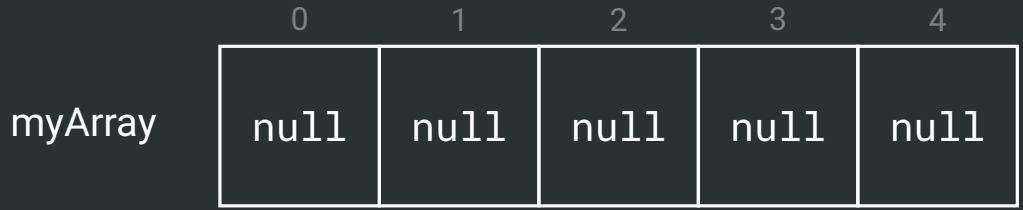
```java
// ArrayIndexOutOfBoundsException on all lines:
System.out.println(myArray[-8]);
System.out.println(myArray[-1]);
System.out.println(myArray[25]);
System.out.println(myArray[400]);
```

Arrays – Basics

Arrays mit Referenztypen

TECHNISCHE UNIVERSITÄT DARMSTADT

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

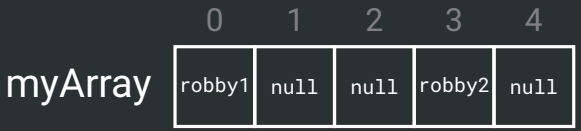


Arrays – Basics

Arrays mit Referenztypen

TECHNISCHE UNIVERSITÄT DARMSTADT

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

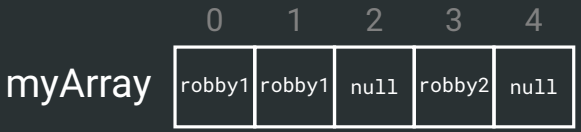


Arrays – Basics

Arrays mit Referenztypen

TECHNISCHE UNIVERSITÄT DARMSTADT

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



Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Organisatorisches

Wiederholung

Arrays – Basics

Eindimmensionale Arrays
- Über Elemente iterieren
- Enhanced for-Loops

Mehrdimensionale Arrays

Eindimmensionale Arrays

Über Elemente iterieren

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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

Eindimmensionale Arrays

Enhanced for-Loops

TECHNISCHE UNIVERSITÄT DARMSTADT

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

```java
for(int value : numArray) {
    System.out.println(value);
}
```

- 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

Eindimmensionale Arrays

Mehrdimensionale Arrays
- Über Elemente iterieren
- Enhanced for

Mehrdimensionale Arrays

TECHNISCHE UNIVERSITÄT DARMSTADT

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

- Zweidimensionale Arrays sind ’Arrays von Arrays’

Mehrdimensionale Arrays

Über Elemente iterieren

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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

Mehrdimensionale Arrays

Über Elemente iterieren

TECHNISCHE UNIVERSITÄT DARMSTADT

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

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

Mehrdimensionale Arrays

Enhanced for

TECHNISCHE UNIVERSITÄT DARMSTADT

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

```java
for(int[] subArray : multiArray) {
    for(int value : subArray) {
        System.out.println(value);
    }
}
```

Live-Coding!