Funktionale und objektorientierte Programmierkonzepte

TECHNISCHE UNIVERSITÄT DARMSTADT

Präsenz-Sprechstunde B

Simon Hock, Nhan Huynh

Arrays

TECHNISCHE UNIVERSITÄT DARMSTADT

- Block von reserviertem Speicher eines gleichen Komponententyps (Datentyp der Elemente)
- Feste Größe
- Formaler Aufbau: `Datentyp[] Bezeichner = new Datentyp[Größe];`
- Kurzform: `Datentyp[] Bezeichner = {Element1, Element2, ...};`

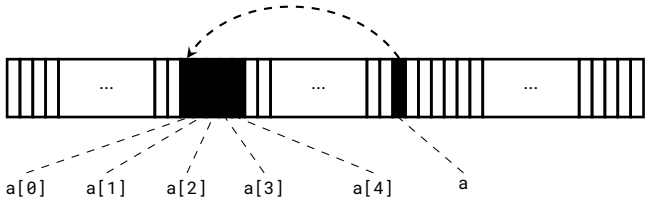

Abbildung: Abstrakte Visualisierung des Speicherplatzes eines Arrays a

Standardwerte

TECHNISCHE UNIVERSITÄT DARMSTADT

Datentyp	Standardwert
`boolean`	`false`
`byte`	`0`
`short`	`0`
`int`	`0`
`long`	`0L`
`float`	`0.0f`
`double`	`0.0d`
`char`	`'\u0000'`
Referenztypen	`null`

Tabelle: Standardwerte für Attribute und Arraykomponenten

Analogie - Bücherregal

TECHNISCHE UNIVERSITÄT DARMSTADT

- Vorstellung Array als Bücherregal
- Größe des Arrays: Anzahl an Fächern
- Fach kann nur ein Buch aufbewahren (eindimensionales Array)

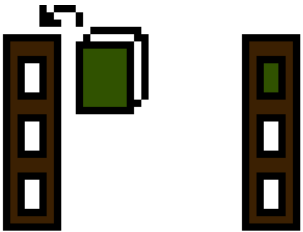

Abbildung: Array als Bücherregal

Klasse Book

```
1  public class Book {
2
3    private final String title;
4    private final String author;
5    private double cost;
6
7    public Book(String title, String author, double cost) {
8      this.title = title;
9      this.author = author;
10     this.cost = cost;
11   }
12   ...
13 }
```

Bücherregal als Array

```
1 Book[] bookshelf = new Book[3];
2 bookshelf[0] = new Book("Der Da Vinci Code", "Dan Brown", 12.0);
3 bookshelf[1] = new Book("Harry Potter und die Heiligtümer des Todes",
4   "Joanne K. Rowling", 13.99);
5 bookshelf[2] = new Book("Der Hobbit", "J. R. R. Tolkien", 18.0);
```

Bücherregal als Array - Kurzform

```
1 Book[] bookshelf = {
2     new Book("Der Da Vinci Code", "Dan Brown", 12.0),
3     new Book("Harry Potter und die Heiligtümer des Todes",
4     "Joanne K. Rowling", 13.99),
5     new Book("Der Hobbit", "J. R. R. Tolkien", 18.0)
6 };
```

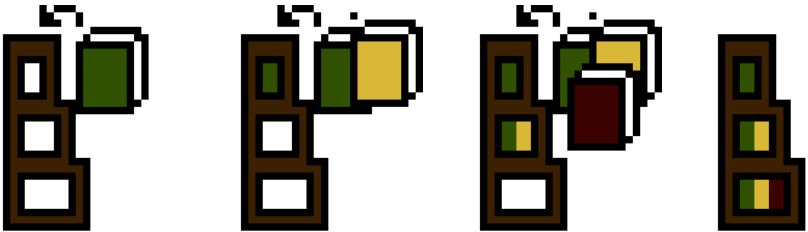

Abbildung: Verschachteltes Array als Bücherregal

Bücherregal als verschachteltes Array

```
1 Book[][] bookshelf = new Book[3][];
2 bookshelf[0] = new Book[1];
3 bookshelf[0][0] = new Book("Der Da Vinci Code", "Dan Brown", 12.0);
4
5 bookshelf[1] = new Book[2];
6 bookshelf[1][0] = new Book("Harry Potter und der Stein der Weisen",
7   "Joanne K. Rowling", 8.99);
8 bookshelf[1][1] = new Book("Harry Potter und die Heiligtümer des Todes",
9   "Joanne K. Rowling", 13.99);
```

Bücherregal als verschachteltes Array

```
11 bookshelf[2][0] = new Book("Der Schmied von Großholzingen",
12   "J. R. R. Tolkien", 14.95);
13 bookshelf[2][1] = new Book("Der Herr der Ringe", "J. R. R. Tolkien", 12.0);
14 bookshelf[2][2] = new Book("Der Hobbit", "J. R. R. Tolkien", 18.0);
```

JUnit - Testen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Framework zum Testen von Java-Programmen
- JUnit 5
- Testen von Methoden auf Korrektheit
- Später Vorführung

JUnit - Importe

TECHNISCHE UNIVERSITÄT DARMSTADT

- `org.junit.Assert`/ `org.junit.api.Assertions`
  - Sammlung von Klassenmethoden zum Testen.
  - `Assert`: Existiert nur noch aus Kompatibilitätsgründen.
  - `Assertions`: Enthält die neueren JUnit 5 Funktionalitäten.
    - `https://junit.org/junit5/docs/5.0.1/api/org/junit/jupiter/api/Assertions.html`
- `org.junit.jupiter.api`
  - Wichtige Funktionalitäten zum Testen bspw. `@Test` Annotation.

JUnit-Annotationen

TECHNISCHE UNIVERSITÄT DARMSTADT

Annotation	Beschreibung
`@Test`	Annotierte Methode ist eine Testmethode.
`@BeforeAll`	Annotierte Methode wird vor allen Tests in der aktuellen Testklasse ausgeführt.
`@AfterAll`	Annotierte Methode wird nach allen Tests in der aktuellen Testklasse ausgeführt.
`@BeforeEach`	Annotierte Methode wird vor jeden Test in der aktuellen Testklasse ausgeführt
`@AfterEach`	Annotierte Methode ist eine Testmethode.

Tabelle: Wichtige Annotationen

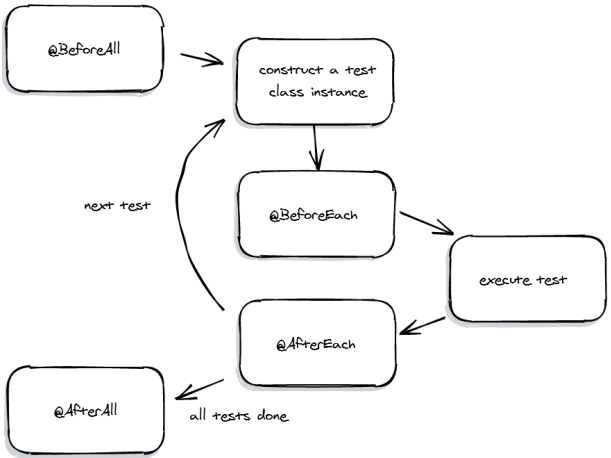

Abbildung: Quelle: https://www.arhohuttunen.com/junit-5-test-lifecycle/

Live Coding

Selbstständiges Arbeiten