

# FOP Recap #9



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

## Exceptions





---

# **Willkommen zurück!**



Primitiven Wert in String umwandeln

Mehrdimensionale Arrays

Exceptions

assert

JUnit-Tests

# Primitiven Wert in String umwandeln



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int value = 0;  
2 String stringValue = value; // ERROR!
```

# Primitiven Wert in String umwandeln



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int value = 0;  
2 String stringValue = "" + value; // OK
```

# Primitiven Wert in String umwandeln



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int value = 0;  
2 String stringValue = String.valueOf(value); // Super
```

# Primitiven Wert in String umwandeln



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int value = 0;  
2 String stringValue = Integer.toString(value); // Exotisch
```



Primitiven Wert in String umwandeln

Mehrdimensionale Arrays

- 2D Initialisierung

- 2D Zugriff auf Elemente

- 2D Beispiel

Exceptions

assert

JUnit-Tests





- Haben als Komponententypen wieder Arrays
- Typischerweise 2D oder 3D Arrays
- Zugriff über mehrere Indices
- Wichtig: Subarrays müssen nicht die gleiche Länge haben
- Zusätzliche Kniffe fürs Erstellen

# Mehrdimensionale Arrays

## 2D Initialisierung



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int[][] bigArray = new int[2][5];
```

```
1 int[][] bigArray = new int[2][];  
2 bigArray[0] = new int[5];  
3 bigArray[1] = new int[5];
```

```
1 int[][] bigArray = new int[2][];  
2 bigArray[0] = new int[] {0, 0, 0, 0, 0};  
3 bigArray[1] = new int[] {0, 0, 0, 0, 0};
```

```
1 int[][] bigArray = new int[][] {{0, 0, 0, 0, 0}, {0, 0, 0, 0, 0}};
```

# Mehrdimensionale Arrays

## 2D Zugriff auf Elemente



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int[][] bigArray = new int[2][5];
```

```
1 int[] smallArray = bigArray[0];  
2 smallArray[2] = 3;
```

```
1 bigArray[0][2] = 3;
```

# Mehrdimensionale Arrays

## 2D Zugriff auf Elemente



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int[][] bigArray = new int[2][5];
```

```
1 int[] smallArray = bigArray[0];
```

```
2 int value = smallArray[2];
```

```
1 int value = bigArray[0][2];
```

# Mehrdimensionale Arrays

## 2D Beispiel



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int[][] array = new int[3][];  
2 array[0] = new int[] {1, 2, 3, 4, 5};  
3 array[1] = new int[] {3, 9};  
4 array[2] = new int[] {5, -11, 8};
```

|          | 0 | 1   | 2 | 3 | 4 |
|----------|---|-----|---|---|---|
| array[0] | 1 | 2   | 3 | 4 | 5 |
| array[1] | 3 | 9   |   |   |   |
| array[2] | 5 | -11 | 8 |   |   |

# Mehrdimensionale Arrays

## 2D Beispiel



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int[][] array = ....;  
2 array[0][0] = 10;  
3 array[2][2] = -10;
```

|          | 0  | 1   | 2   | 3 | 4 |
|----------|----|-----|-----|---|---|
| array[0] | 10 | 2   | 3   | 4 | 5 |
| array[1] | 3  | 9   |     |   |   |
| array[2] | 5  | -11 | -10 |   |   |

# Mehrdimensionale Arrays

## 2D Beispiel



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 int[][] array = new int[3][];  
2 array[1] = new int[] {3, 9, 10, 25};
```

|          |      |   |    |    |
|----------|------|---|----|----|
| array[0] | null |   |    |    |
| array[1] | 3    | 9 | 10 | 25 |
| array[2] | null |   |    |    |



Primitiven Wert in String umwandeln

Mehrdimensionale Arrays

Exceptions

- Einfaches Beispiel

- Besseres Beispiel

- Mehrere Exceptions

- Falsche Reihenfolge der Catch-Blöcke

- Korrekte Reihenfolge der Catch-Blöcke

- Mehrere Exceptions

- Weiterreichen

- Ausnahme RuntimeException

Javadoc

Typische Fehler





- Grobe Unterscheidung zwischen
  - ▣ Fehlern
  - ▣ Laufzeitfehlern
- In Java: Viele Fehler bereits beim Kompilieren erkennbar
- Laufzeitfehler sind:
  - ▣ Fehler die erst während der Laufzeit auftreten
  - ▣ Fehler, die situationsbedingt auftreten können
- Bereits bekannte Laufzeitfehler:
  - ▣ NullPointerException
  - ▣ ArrayIndexOutOfBoundsException
  - ▣ ClassCastException
  - ▣ IOException
  - ▣ ....
- Erben alle direkt oder indirekt von `Exception`

# Exceptions

## Einfaches Beispiel



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 public int buyCoolHat(int amountOfMoney) throws Exception {  
2     if(amountOfMoney < 50) {  
3         throw new Exception("Insufficient funds!");  
4     }  
5  
6     System.out.println("Good choice.");  
7     return amountOfMoney - 50;  
8 }
```

# Exceptions

## Einfaches Beispiel



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 public int buyCoolHat(int amountOfMoney) throws Exception {  
2     // ....  
3 }
```

```
1 int myMoney = 70;  
2  
3 try {  
4     myMoney = buyCoolHat(myMoney);  
5 }  
6 catch(Exception e) {  
7     e.printStackTrace();  
8 }
```

# Exceptions

## Einfaches Beispiel



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 public int buyCoolHat(int amountOfMoney) throws  
  ↳ NullPointerException {  
2     throw new NullPointerException("Not available!");  
3 }
```

```
1 int myMoney = 70;  
2  
3 try {  
4     myMoney = buyCoolHat(myMoney);  
5 }  
6 catch(NullPointerException e) {  
7     e.printStackTrace();  
8 }
```

# Exceptions

## Besseres Beispiel



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 public int buyCoolHat(int amountOfMoney) throws
   ↳ InsufficientFundsException {
2     // ....
3 }
```

```
1 int myMoney = 70;
2
3 try {
4     myMoney = buyCoolHat(myMoney);
5 }
6 catch (InsufficientFundsException e) {
7     e.printStackTrace();
8 }
```

# Exceptions

## Besseres Beispiel



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 public int buyCoolHat(int amountOfMoney) throws
   ↳ InsufficientFundsException {
2     if(amountOfMoney < 50) {
3         throw new InsufficientFundsException(50 - amountOfMoney);
4     }
5     // ....
6 }
```

```
1 public class InsufficientFundsException extends Exception {
2     public InsufficientFundsException(int missingMoney) {
3         super("Insufficient funds! Needed " + missingMoney + "
   ↳ more money.");
4     }
5 }
```

# Exceptions

## Mehrere Exceptions

```
1 public void test() throws NullPointerException,  
   ↪ ClassCastException {  
2     // ....  
3 }
```

```
1 try {  
2     test();  
3 }  
4 catch(NullPointerException e) {  
5     e.printStackTrace();  
6 }  
7 catch(ClassCastException e) {  
8     e.printStackTrace();  
9 }
```

# Exceptions

## Mehrere Exceptions

```
1 public void test() throws NullPointerException,  
   ↳ ClassCastException {  
2     // ....  
3 }
```

```
1 try {  
2     test();  
3 }  
4 catch(NullPointerException | ClassCastException e) {  
5     e.printStackTrace();  
6 }
```



# Exceptions

## Mehrere Exceptions

```
1 public void test() throws NullPointerException,  
   ↪ ClassCastException {  
2     // ....  
3 }
```

```
1 try {  
2     test();  
3 }  
4 catch(Exception e) {  
5     e.printStackTrace();  
6 }
```

# Exceptions

## Falsche Reihenfolge der Catch-Blöcke



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 public void test() throws NullPointerException,  
   ↪ ClassCastException {  
2     // ....  
3 }
```

```
1 try {  
2     test();  
3 }  
4 catch(Exception e) {  
5     e.printStackTrace();  
6 }  
7 catch(NullPointerException e) {  
8     System.out.println("Null!"); // !!!  
9 }
```

# Exceptions

## Korrekte Reihenfolge der Catch-Blöcke



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 public void test() throws NullPointerException,  
   ↪ ClassCastException {  
2     // ....  
3 }
```

```
1 try {  
2     test();  
3 }  
4 catch(NullPointerException e) {  
5     System.out.println("Null!");  
6 }  
7 catch(Exception e) {  
8     e.printStackTrace();  
9 }
```

# Exceptions

## Mehrere Exceptions

```
1 public void test() throws Exception {  
2     throw new NullPointerException("Message");  
3 }
```

```
1 try {  
2     test();  
3 }  
4 catch(NullPointerException | ClassCastException e) {  
5     e.printStackTrace();  
6 }  
7 catch(Exception e) {  
8     System.out.println("You need this :)");  
9 }
```



- Weiterreichen/Weiterleiten von Exceptions ist (unter anderem) dann sinnvoll, wenn
  - ▣ der Fehler nicht an dieser Stelle gelöst werden kann
  - ▣ es eine höhere zentrale Stelle zum Sammeln von Fehlern gibt

# Exceptions

## Weiterreichen



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 public void test() throws NullPointerException,  
   ↳ ClassCastException {  
2     throw new NullPointerException("Message");  
3 }
```

```
1 public void run() throws ClassCastException {  
2     try {  
3         test();  
4     }  
5     catch(NullPointerException e) { }  
6     catch(ClassCastException e) {  
7         throw e;  
8     }  
9 }
```

# Exceptions

## Weiterreichen



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 public void test() throws NullPointerException,  
   ↳ ClassCastException {  
2     throw new NullPointerException("Message");  
3 }
```

```
1 public void run() throws ClassCastException {  
2     // Kein try und catch für ClassCastException  
3     try {  
4         test();  
5     }  
6     catch(NullPointerException e) {  
7         e.printStackTrace();  
8     }  
9 }
```



- ACHTUNG! Ausnahme!
- Alle Exceptions die von RuntimeException direkt oder indirekt erben
  - ▣ Müssen nicht mit throws deklariert werden
  - ▣ Müssen nicht mit einem try-and-catch Block abgeprüft werden
- Hierzu zählen zum Beispiel
  - ▣ NullPointerException
  - ▣ ArrayIndexOutOfBoundsException
  - ▣ ClassCastException
  - ▣ ....



# Exceptions

## Javadoc



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1  /**
2   * Buys a cool hat
3   * @param amountOfMoney The amount of money available
4   * @return Amount of money left after transaction
5   * @throws InsufficientFundsException If there is not enough money
6   *         ↪ available
7   */
8  public int buyCoolHat(int amountOfMoney) throws
9   ↪ InsufficientFundsException {
10     if(amountOfMoney < 50) {
11         throw new InsufficientFundsException(50 - amountOfMoney);
12     }
13     return amountOfMoney - 50;
14 }
```

# Exceptions

## Typische Fehler



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 public class X {  
2     public static int calculateTheAnswer() throws  
    ↳ IllegalStateException {  
3         if(Y.complexCondition() == false) {  
4             throw new IllegalStateException("Nobody knows why  
                ↳ this failed.");  
5         }  
6         return 42;  
7     }  
8 }
```

# Exceptions

## Typische Fehler



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 public static int tryToCall() {  
2     // ....  
3 }
```

# Exceptions

## Typische Fehler



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 public static int tryToCall() {  
2     try {  
3         int returnValue = X.calculateTheAnswer();  
4     }  
5     catch(IllegalStateException e) {  
6         throw new RuntimeException("Look at this: " +  
           ↪ e.getMessage());  
7     }  
8     return returnValue; // ERROR!  
9 }
```

# Exceptions

## Typische Fehler



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 public static int tryToCall() {  
2     int returnValue;  
3     try {  
4         returnValue = X.calculateTheAnswer();  
5     }  
6     catch(IllegalStateException e) {  
7         throw new RuntimeException("Look at this: " +  
8             ↪ e.getMessage());  
9     }  
10    return returnValue;  
}
```

# Exceptions

## Typische Fehler



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 public static int tryToCall() {  
2     try {  
3         return X.calculateTheAnswer();  
4     }  
5     catch(IllegalStateException e) {  
6         throw new RuntimeException("Look at this: " +  
           ↪ e.getMessage());  
7     }  
8 }
```

- ExceptionA erbt direkt oder indirekt von Exception
- ExceptionB erbt direkt oder indirekt von ExceptionA

```
1 public class A {  
2     public void myMethod() throws ExceptionA {  
3         // ...  
4     }  
5 }
```

# Exceptions

## Vererbung



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 public class B extends A {  
2     @Override  
3     public void myMethod() throws ExceptionA {  
4         // Gleicher Exception-Typ erlaubt  
5     }  
6 }
```

```
1 public class C extends A {  
2     @Override  
3     public void myMethod() throws ExceptionB {  
4         // "Präzisierung" erlaubt  
5     }  
6 }
```



# Das steht heute auf dem Plan



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Primitiven Wert in String umwandeln

Mehrdimensionale Arrays

Exceptions

**assert**

JUnit-Tests



- Einfache und kurze Möglichkeit um Fehler zu werfen
- Wirft immer eine `AssertionError`, der von `Error` abgeleitet ist
- `Error` erbt weder von `Exception` noch `RuntimeException`
- Ist im Normalfall deaktiviert!
- Muss mittels `-enableassertions` bzw `-ea` aktiviert werden

```
1 public static void dummyTest(int b) {  
2     assert b >= 0: "Argument must be positive!";  
3 }
```

```
1 public static int sum(int a, int b) {  
2     assert ((long)a + (long)b) >= Integer.MIN_VALUE &&  
3         ((long)a + (long)b) <= Integer.MAX_VALUE: "Overflow  
    ↪ detected!";  
4     return a + b;  
5 }
```



**assert**



**Exceptions**

# Das steht heute auf dem Plan



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

Primitiven Wert in String umwandeln

Mehrdimensionale Arrays

Exceptions

`assert`

JUnit-Tests

Einfaches Beispiel

`assertDoesNotThrow`

`assertThrowsExactly`



- Framework um einfache aber auch sehr komplexe Tests für Java-Programme zu schreiben
- Library muss erst eingebunden werden
- Werden auch in Java geschrieben

# JUnit-Tests

## Einfaches Beispiel



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

```
1 import org.junit.jupiter.api.Test;
2 import static org.junit.jupiter.api.Assertions.*;
3
4 public class StudentTests {
5     ....
6 }
```



```
1  @Test
2  public void myTest() {
3      Tree tree = new LemonTree();
4      Apple unexpectedFruit = new Apple(5);
5
6      assertTrue(tree.hasFruit());
7      assertTrue(tree.getFruit() instanceof Lemon);
8      assertNotEquals(unexpectedFruit, tree.getFruit());
9
10     assertEquals(2, tree.getFruitAmount());
11 }
```





```
1  @Test
2  public void myTest() {
3      HatFactory factory = new HatFactory();
4
5
6      int moneyLeft = assertDoesNotThrow(
7          () -> factory.buyCoolHat(52)
8      );
9      assertEquals(moneyLeft, 2, "Incorrect money left!");
10 }
```



```
1  @Test
2  public void myTest() {
3      HatFactory factory = new HatFactory();
4
5
6      Throwable thrownException = assertThrowsExactly(
7          InsufficientFundsException.class,
8          () -> factory.buyCoolHat(49)
9      );
10     String message = thrownException.getMessage();
11     // assertEquals....
12 }
```



---

# Live-Coding!