

JUnit

Konrad Raue, Oliver Scholz

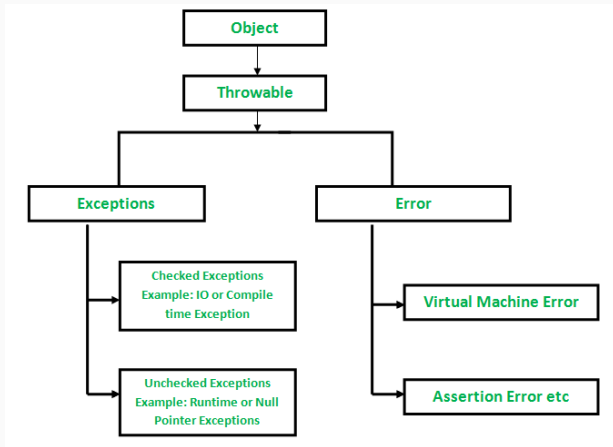
17. Dezember 2019

1. Exception
2. Try-Catch
3. UnitTest
4. Nächste Woche

Exception

Exceptions

- unerwartetes Event
- sollen versucht werden Aufzufangen (im Gegensatz zum Error)
- sind Klassen welche von Exception erben
- Unterteilung in checked und unchecked



eigene Exception schreiben

```
1 public class MeineException extends Exception{  
2     public MeineException(){  
3         super();  
4     }  
5  
6     public MeineException(String message){  
7         super(message);  
8     }  
9 }
```

Es gibt aber auch schon einige Vorgefertigte wie die `NullPointerException` oder die `IllegalStateException`.

eigene Exception verwenden

```
1 public class Verwendung{
2     public static void main(String[] args) throws Exception{
3         int a = 1;
4         int b = 4;
5         if(a != b){
6             throw new MeineException("a und b sind ungleich");
7         }
8     }
9 }
```

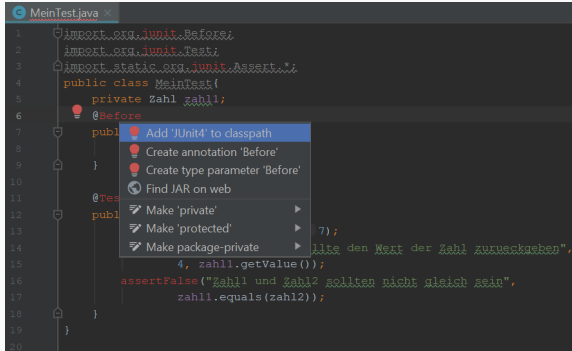
Try-Catch

try-catch

```
1 public class Verwendung {
2     public static void main(String[] args){
3         int a = 2;
4         int b = 3;
5         try{
6             if(args.length < 2){
7                 throw new NullPointerException();
8             }
9             a = Integer.parseInt(args[0]);
10            b = Integer.parseInt(args[1]);
11            if(a != b){
12                throw new MeineException();
13            }
14        }catch(NullPointerException nullPointerException){
15            a = 0;
16            b = 0;
17        }catch(MeineException meineException){
18            b = a;
19        }finally{
20            System.out.println("a=" + a + ", b=" + b);
21        }
22    }
23    System.out.println("Fertig");
24 }
25 }
```


UnitTest

- Klasse, die unter Verwendung von JUnit ausführbar ist
- org.junit muss importiert werden
- JUnit4 oder JUnit5 mit unterschiedlicher Implementierung
- Annotations (@Test) oder Benennungen (Methoden heißen test_())
- assertEquals(Grund:String(optional), erwarteterWert, wirklicherWert);
- assertEquals(), assertTrue(), assertFalse(), assertNull(), assertNotNull() uvm.
- <https://junit.org/junit5/>



```

1  import org.junit.Before;
2  import org.junit.Test;
3  import static org.junit.Assert.*;
4  public class MeinTest{
5      private Zahl zahl1;
6      @Before
7      publ
8
9
10
11     @Tes
12     publ
13
14
15         4, zahl1.getValue());
16         assertEquals("Zahl1 und Zahl2 sollten nicht gleich sein",
17             zahl1.equals(zahl2));
18
19 }
20

```

UnitTest in JUnit4

```
1 import org.junit.Before;
2 import org.junit.Test;
3 import static org.junit.Assert.*;
4 public class MeinTest{
5     private Zahl zahl1;
6     @Before
7     public void setup(){
8         zahl1 = new Zahl(4);
9     }
10
11     @Test
12     public void test1(){
13         Zahl zahl2 = new Zahl(7);
14         assertEquals("getValue() sollte den Wert der Zahl
15             zurueckgeben", 4, zahl1.getValue());
16         assertFalse("Zahl1 und Zahl2 sollten nicht gleich sein",
17             zahl1.equals(zahl2));
18     }
19
20     @Test
21     public void test2(){
22         ...
23     }
24     ...
25 }
```

UnitTest in JUnit4 mit TestCase

```
1 import junit.framework.TestCase;
2
3 public class MeinTest extends TestCase{
4     private Zahl zahl1;
5
6     public void setUp(){
7         zahl1 = new Zahl(4);
8     }
9
10    public void testValue(){
11        Zahl zahl2 = new Zahl(7);
12        assertEquals("getValue() sollte den Wert der Zahl
13            zurueckgeben", 4, zahl1.getValue());
14        assertFalse("Zahl1 und Zahl2 sollten nicht gleich sein",
15            zahl1.equals(zahl2));
16    }
17
18    public void testAnderes(){
19        ...
20    }
21    ...
22 }
```

UnitTest in JUnit5

```
1 import org.junit.jupiter.api.BeforeAll;
2 import org.junit.jupiter.api.Test;
3 import org.junit.jupiter.api.TestInstance;
4 import static org.junit.jupiter.api.Assertions.*;
5 @TestInstance(TestInstance.Lifecycle.PER_CLASS)
6 public class MeinTest{
7     private Zahl zahl1;
8     @BeforeAll
9     public void setup(){
10         zahl1 = new Zahl(4);
11     }
12
13     @Test
14     public void test1(){
15         Zahl zahl2 = new Zahl(7);
16         assertEquals(4, zahl1.getValue());
17         assertFalse(zahl1.equals(zahl2));
18     }
19
20     @Test
21     public void test2(){
22         ...
23     }
24     ...
25 }
```

Nächste Woche

- UML