Teil VI

Ausnahmebehandlung

Throwable

```
java.lang.Object
 ⌞ java.lang.Throwable
   ⌞ java.lang.Error
     ⌞ java.lang.VirtualMachineError
       ⌞ java.lang.OutOfMemoryError
   ⌞ java.lang.Exception
     ⌞ java.lang.RuntimeException
       ⌞ java.lang.NullPointerException
                   ArithmeticException        z.B. Integer Division by Zero
```

Nur die Konstruktoren sind überladen.

Throwable

- `Throwable ( String message )`
- `String getMessage()`
- `void printStackTrace()`
- `String toString()`

Throwable fangen

```
1 try {
2     <statements>
3 } catch ( Throwable thr ) {
4     <tue etwas sinnvolles>
5 } finally {
6     <aufraeumen>
7 }
```

Throwable fangen

Mehrere Throwable fangen

- von speziell zu allgemein

Throwable fangen

Mehrere Throwable fangen

- von speziell zu allgemein

Beispiel:

1. NullPointerException
2. RuntimeException
3. Exception
4. Throwable

Eigenes Throwable erstellen

```
1  public class MyException
2      extends Exception {
3
4      public MyException () {
5          super();
6      }
7
8      public MyException ( String message ) {
9          super( message );
10     }
11 }
```

Eigenes Throwable werfen und fangen

```
1  private int myMethod (
2      int a
3  ) throws MyException {
4      if ( a<0 ) {
5          throw new MyException( "..." );
6      } else
7          return Math.sqrt(a);
8      }
9  }
```

Eigenes Throwable werfen und fangen

```java
private int myMethod (
    int a
) throws MyException {
    if ( a<0 ) {
        throw new MyException( "..." );
    } else
        return Math.sqrt(a);
    }
}
```

```java
private int myComputation (
    int b
) {
    try {
        wurzel = myMethod(b);
    } catch (MyException ex) {
        wurzel = - myMethod(-b);
    }
}
```

$$b = 5 \quad \rightarrow \qquad\qquad\qquad\quad \rightarrow \quad wurzel = \sqrt{5}$$
$$b = -5 \quad \rightarrow \quad MyException \quad \rightarrow \quad wurzel = -\sqrt{-5}$$