# Das java.lang-Package

Die Klasse [java.lang.Object](https://www.youtube.com/watch?v=BLz_YfgbmmQ) ist automatisch/implizit die Superklasse aller Klassen, wenn diese nicht anderweitig erbt. Selbst dann ist aber deren Superklasse irgendwann allein.  
Das bedeutet, dass alle Klassenhierarchien nach oben die in die Object-Klasse münden.

Hier ein Auszug aus der [JavaSE8-Doc](https://docs.oracle.com/javase/8/docs/api/java/lang/package-summary.html) mit den mir bisher bekannten Elementen (zitiert):

**Interfaces** (angekratzt, weniger wissend (kurz eingelesen), aber damit zu tun gehabt)

|  |  |
| --- | --- |
| [Cloneable](https://docs.oracle.com/javase/8/docs/api/java/lang/Cloneable.html)  Ich hatte durch  Arrays damit zu tun. | A class implements the Cloneable interface to indicate to the [Object.clone()](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html" \l "clone--) method that it is legal for that method to make a field-for-field copy of instances of that class. |
| [Comparable](https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html)<T> Durch compareTo() | This interface imposes a total ordering on the objects of each class that implements it. |
| [Iterable](https://docs.oracle.com/javase/8/docs/api/java/lang/Iterable.html)<T> Durch Prüfungstest | Implementing this interface allows an object to be the target of the "for-each loop" statement. (Dies will ich noch nachschauen). |

**Klassen** (Kennengelernt oder wissentlich schon verwendet, auch nur auszugsweise)

|  |  |
| --- | --- |
| [Boolean](https://docs.oracle.com/javase/8/docs/api/java/lang/Boolean.html) | The Boolean class wraps a value of the primitive type boolean in an object. |
| [Byte](https://docs.oracle.com/javase/8/docs/api/java/lang/Byte.html) | The Byte class wraps a value of primitive type byte in an object. |
| [Character](https://docs.oracle.com/javase/8/docs/api/java/lang/Character.html) | The Character class wraps a value of the primitive type char in an object. |
| [Double](https://docs.oracle.com/javase/8/docs/api/java/lang/Double.html) | The Double class wraps a value of the primitive type double in an object. |
| [Float](https://docs.oracle.com/javase/8/docs/api/java/lang/Float.html) | The Float class wraps a value of primitive type float in an object. |
| [Long](https://docs.oracle.com/javase/8/docs/api/java/lang/Long.html) | The Long class wraps a value of the primitive type long in an object. |
| [Math](https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html) | The class Math contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions. |
| [Number](https://docs.oracle.com/javase/8/docs/api/java/lang/Number.html) | The abstract class Number is the superclass of platform classes representing numeric values that are convertible to the primitive types byte, double, float, int, long, and short. (Ermöglicht das Casting implizit und explizit, wenn ich das richtig verstehe.) |
| [Object](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html) | Class Object is the root of the class hierarchy. |
| [SecurityManager](https://docs.oracle.com/javase/8/docs/api/java/lang/SecurityManager.html) | The security manager is a class that allows applications to implement a security policy. -Noch bekannt aus meinem Vortrag |
| [Short](https://docs.oracle.com/javase/8/docs/api/java/lang/Short.html) | The Short class wraps a value of primitive type short in an object. |
| [StackTraceElement](https://docs.oracle.com/javase/8/docs/api/java/lang/StackTraceElement.html) | An element in a stack trace, as returned by [Throwable.getStackTrace()](https://docs.oracle.com/javase/8/docs/api/java/lang/Throwable.html" \l "getStackTrace--). (Die Fehlerspur einer Exception) |
| [String](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html) | The String class represents character strings. |
| [StringBuilder](https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html) | A mutable sequence of characters. (Bisher nur vage verwendet) |
| [System](https://docs.oracle.com/javase/8/docs/api/java/lang/System.html) | The System class contains several useful class fields and methods. (Wir nutzen die PrintStream-Variable out, um die Methoden print, println, printf und format davon zu gebrauchen.) |
| [Throwable](https://docs.oracle.com/javase/8/docs/api/java/lang/Throwable.html) | The Throwable class is the superclass of all errors and exceptions in the Java language. (Mutter aller Exceptions) |
| [Void](https://docs.oracle.com/javase/8/docs/api/java/lang/Void.html) | The Void class is an uninstantiable placeholder class to hold a reference to the Class object representing the Java keyword void. (Interessant zu Wissen, dass es einen Platzhalter gibt.) |

Zusammenfassend kenne ich also: Alle Wrapper-Klassen, String, System(.out), Stringbuilder (nur lose), Throwable (als Exceptions) und Math.

**Exceptiones** (auch nur vage, aber einmal zum Erinnern, was ich schon kenne)

|  |  |
| --- | --- |
| [ArithmeticException](https://docs.oracle.com/javase/8/docs/api/java/lang/ArithmeticException.html) | Thrown when an exceptional arithmetic condition has occurred. |
| [ArrayIndexOutOfBoundsException](https://docs.oracle.com/javase/8/docs/api/java/lang/ArrayIndexOutOfBoundsException.html) | Thrown to indicate that an array has been accessed with an illegal index. |
| [Exception](https://docs.oracle.com/javase/8/docs/api/java/lang/Exception.html) | The class Exception and its subclasses are a form of Throwable that indicates conditions that a reasonable application might want to catch. |
| [IllegalArgumentException](https://docs.oracle.com/javase/8/docs/api/java/lang/IllegalArgumentException.html) | Thrown to indicate that a method has been passed an illegal or inappropriate argument. |
| [IndexOutOfBoundsException](https://docs.oracle.com/javase/8/docs/api/java/lang/IndexOutOfBoundsException.html) | Thrown to indicate that an index of some sort (such as to an array, to a string, or to a vector) is out of range. |
| [NegativeArraySizeException](https://docs.oracle.com/javase/8/docs/api/java/lang/NegativeArraySizeException.html) | Thrown if an application tries to create an array with negative size. |
| [NoSuchFieldException](https://docs.oracle.com/javase/8/docs/api/java/lang/NoSuchFieldException.html) | Signals that the class doesn't have a field of a specified name. |
| [NoSuchMethodException](https://docs.oracle.com/javase/8/docs/api/java/lang/NoSuchMethodException.html) | Thrown when a particular method cannot be found. |
| [NullPointerException](https://docs.oracle.com/javase/8/docs/api/java/lang/NullPointerException.html) | Thrown when an application attempts to use null in a case where an object is required. |
| [NumberFormatException](https://docs.oracle.com/javase/8/docs/api/java/lang/NumberFormatException.html) | Thrown to indicate that the application has attempted to convert a string to one of the numeric types, but that the string does not have the appropriate format. |
| [RuntimeException](https://docs.oracle.com/javase/8/docs/api/java/lang/RuntimeException.html) | RuntimeException is the superclass of those exceptions that can be thrown during the normal operation of the Java Virtual Machine. |
| [SecurityException](https://docs.oracle.com/javase/8/docs/api/java/lang/SecurityException.html) | Thrown by the security manager to indicate a security violation. |
| [StringIndexOutOfBoundsException](https://docs.oracle.com/javase/8/docs/api/java/lang/StringIndexOutOfBoundsException.html) | Thrown by String methods to indicate that an index is either negative or greater than the size of the string. |

# Sichtbarkeit und Lebensspanne von Variablen

Aus der Hausaufgabe für die Ferien ging aus meiner Recherche hervor:

**Variablen auf Instanz- und Klassenebene** (also ~~static/block und~~ field/member-Variablen) **müssen nicht explizit initialisiert werden**, sondern werden implizit mit ihrem Standardwert belegt, sofern sie nicht anderweitig initialisiert werden.

**Variablen innerhalb von Methoden hingegen müssen explizit initialisiert werden.**

Dies möchte ich nach theoretischer Erarbeitung nochmal versuchen praktisch gegenzuprüfen, indem ich eine kleine Testklasse schreibe und alle Fälle durchprobiere.

Dazu werde ich in jedem benannten Scope zwei Variablen zur Gegenprobe anlegen und dann versuchen diese wieder auszulesen. Die IDE dürfte zuvor schon was rückmelden.

Des Weiteren will ich die Sichtbarkeit prüfen, indem ich eine weitere Methode anlege und versuche innerhalb dieser die aus der Klasse zu lesen und umgekehrt.

Für die Member und Blockvariablen, soll es noch eine Testinstanz der Klasse geben.

Zu testen sind nun also jeweils einmal vorbelegt und einmal leergelassen:

statisch, statischer Block, Initialblock der Instanz, Instanz und in einer Methode.

Ich werde alle vom Typ int anlegen. Alle expliziten mit 1, implizit wird 0 verbleiben.

**Erwartete Ergebnisse:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Scope** | **unbelegt** | **Ergebnis** | **belegt** | **Ergebnis** | **Sichtbarkeit** |
| static | sLeer | 0 | sVoll | 1 | Klasse |
| static block | sbLeer | 0 | sbVoll | 1 | Klasse |
| init block | ibLeer | 0 | ibVoll | 1 | Klasse |
| instance | iLeer | 0 | iVoll | 1 | Instanz |
| methode | mLeer | Fehler | mVoll | 1 | Methode |

**Beobachtete Ergebnisse:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Scope** | **unbelegt** | **Ergebnis** | **belegt** | **Ergebnis** | **Sichtbarkeit** |
| static | sLeer | 0 | sVoll | 1 | Klasse, Instanz |
| static block | sbLeer | Fehler | sbVoll | 1 | nur im Block |
| init block | ibLeer | Fehler | ibVoll | 1 | nur im Block |
| instance | iLeer | 0 | iVoll | 1 | Instanz |
| methode | mLeer | Fehler | mVoll | 1 | Methode |

Also eine implizite Initialisierung innerhalb von Blockanweisungen kann ich so nicht bestätigen. Alles andere scheint bzgl. Lebensspanne und Initialisierung zu stimmen.