Java Operatoren

Simple Assignment Operator

```
=       Simple assignment operator
```

Arithmetic Operators

```
+       Additive operator (also used for String concatenation)
-       Subtraction operator
*       Multiplication operator
/       Division operator
%       Remainder operator
```

Unary Operators

```
+       Unary plus operator; indicates positive value (numbers are
        positive without this, however)
-       Unary minus operator; negates an expression
++      Increment operator; increments a value by 1
--      Decrement operator; decrements a value by 1
!       Logical complement operator; inverts the value of a boolean
```

Equality and Relational Operators

```
==      Equal to
!=      Not equal to
>       Greater than
>=      Greater than or equal to
<       Less than
<=      Less than or equal to
```

Conditional Operators

```
&&      Conditional-AND
||      Conditional-OR
?:      Ternary (shorthand for if-then-else statement)
```

Type Comparison Operator

```
instanceof      Compares an object to a specified type
```

Exceptions

Throwing an Exception of Type *Type*

To throw an Exception of Type *Type* (e.g. *Type*=IllegalArgumentException) write the following:

```
throw new Type();

or

throw new Type("Message to report.");
```

Javadoc Auszug

Gegebenenfalls für die Prüfung relevante Auszüge aus der Java 7 Javadoc. Falls Sie mehr Details benötigen um eine Methode zu verwenden, kontaktieren Sie eine Prüfungsaufsicht. Dies sollte in der Regel nicht notwendig sein.

Klasse ArrayList:

ArrayList()
Constructs an empty list with an initial capacity of ten.

ArrayList(Collection<? extends E> c)
Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.

ArrayList(int initialCapacity)
Constructs an empty list with the specified initial capacity.

Modifier and Type	Method and Description
boolean	**add**(E e) Appends the specified element to the end of this list.
void	**add**(int index, E element) Inserts the specified element at the specified position in this list.
void	**clear**() Removes all of the elements from this list.
boolean	**contains**(Object o) Returns `true` if this list contains the specified element.
E	**get**(int index) Returns the element at the specified position in this list.
int	**indexOf**(Object o) Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
boolean	**isEmpty**() Returns `true` if this list contains no elements.
Iterator<E>	**iterator**() Returns an iterator over the elements in this list in proper sequence.
int	**lastIndexOf**(Object o) Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
E	**remove**(int index) Removes the element at the specified position in this list.
boolean	**remove**(Object o) Removes the first occurrence of the specified element from this list, if it is present.
E	**set**(int index, E element) Replaces the element at the specified position in this list with the specified element.
int	**size**() Returns the number of elements in this list.

Klasse HashMap

HashMap()
Constructs an empty HashMap with the default initial capacity 16 and the default load factor 0.75.
HashMap(int initialCapacity)
Constructs an empty HashMap with the specified initial capacity and the default load factor (0.75).

Modifier and Type	Method and Description
void	**clear**() Removes all of the mappings from this map.
boolean	**containsKey**(Object key) Returns true if this map contains a mapping for the specified key.
boolean	**containsValue**(Object value) Returns true if this map maps one or more keys to the specified value.
Set<Map.Entry<K,V>>	**entrySet**() Returns a Set view of the mappings contained in this map.
V	**get**(Object key) Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
boolean	**isEmpty**() Returns true if this map contains no key-value mappings.
Set<K>	**keySet**() Returns a Set view of the keys contained in this map.
V	**put**(K key, V value) Associates the specified value with the specified key in this map.
void	**putAll**(Map<? extends K,? extends V> m) Copies all of the mappings from the specified map to this map.
V	**remove**(Object key) Removes the mapping for the specified key from this map if present.
int	**size**() Returns the number of key-value mappings in this map.
Collection<V>	**values**() Returns a Collection view of the values contained in this map.

Interface Iterator

Modifier and Type	Method and Description
boolean	**hasNext**() Returns true if the iteration has more elements.
E	**next**() Returns the next element in the iteration.
Void	**remove**() Removes from the underlying collection the last element returned by this iterator (optional operation). This method can be called only once per call to **next()**. The behavior of an iterator is unspecified if the underlying collection is modified while the iteration is in progress in any way other than by calling this method. Throws: UnsupportedOperationException - if the remove operation is not supported by this iterator

IllegalStateException - if the next method has not yet been called, or the remove method has already been called after the last call to the next method

Klasse HashSet

HashSet()
Constructs a new, empty set; the backing HashMap instance has default initial capacity (16) and load factor (0.75).

HashSet(Collection<? extends E> c)
Constructs a new set containing the elements in the specified collection.

HashSet(int initialCapacity)
Constructs a new, empty set; the backing HashMap instance has the specified initial capacity and default load factor (0.75).

Modifier and Type	Method and Description
boolean	**add**(E e) Adds the specified element to this set if it is not already present.
void	**clear**() Removes all of the elements from this set.
Object	**clone**() Returns a shallow copy of this HashSet instance: the elements themselves are not cloned.
boolean	**contains**(Object o) Returns true if this set contains the specified element.
boolean	**isEmpty**() Returns true if this set contains no elements.
Iterator<E>	**iterator**() Returns an iterator over the elements in this set.
boolean	**remove**(Object o) Removes the specified element from this set if it is present.
int	**size**() Returns the number of elements in this set (its cardinality).

Klasse Math

Modifier and Type	Method and Description
static long	**round(double a)** Returns the closest long to the argument, with ties rounding to positive infinity.
static int	**round(float a)** Returns the closest int to the argument, with ties rounding to positive infinity.
static double	**random**() Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.

Note: Valid values for type are: int, long, float, double

static *type*	**abs**(*type* a)

	Returns the absolute value of `a` of type *`type`*.
`static type`	**`max`**`(type a, type b)` Returns the larger of the two values `a` and `b` of type *`type`*
`static type`	**`min`**`(type a, type b)` Returns the smaller of the two values a and `b` of type *`type`*

Klasse String

`String`()
Initializes a newly created `String` object so that it represents an empty character sequence.

`String`(`String` original)
Initializes a newly created `String` object so that it represents the same sequence of characters as the argument; in other words, the newly created string is a copy of the argument string.

Modifier and Type	Method and Description
`char`	`charAt(int index)` Returns the char value at the specified index.
`int`	compareTo(String anotherString) Compares two strings lexicographically.
`int`	compareToIgnoreCase(String str) Compares two strings lexicographically, ignoring case differences.
`boolean`	contains(CharSequence s) Returns true if and only if this string contains the specified sequence of char values.
`boolean`	endsWith(String suffix) Tests if this string ends with the specified suffix.
`boolean`	equals(Object anObject) Compares this string to the specified object.
`boolean`	equalsIgnoreCase(String anotherString) Compares this String to another String, ignoring case considerations.
`static String`	format(Locale l, String format, Object... args) Returns a formatted string using the specified locale, format string, and arguments.
`static String`	format(String format, Object... args) Returns a formatted string using the specified format string and arguments.
`int`	indexOf(String str) Returns the index within this string of the first occurrence of the specified substring.
`int`	indexOf(String str, int fromIndex) Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.
`boolean`	isEmpty() Returns true if, and only if, length() is 0.
`static String`	join(CharSequence delimiter, CharSequence... elements) Returns a new String composed of copies of the CharSequence elements joined together with a copy of the specified delimiter.
`int`	lastIndexOf(String str) Returns the index within this string of the last occurrence of the specified substring.
`int`	lastIndexOf(String str, int fromIndex)

	Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index.
`int`	length() Returns the length of this string.
`boolean`	matches(String regex) Tells whether or not this string matches the given regular expression.
`String`	replace(CharSequence target, CharSequence replacement) Replaces each substring of this string that matches the literal target sequence with the specified literal replacement sequence.
`String`	replaceAll(String regex, String replacement) Replaces each substring of this string that matches the given regular expression with the given replacement.
`String`	replaceFirst(String regex, String replacement) Replaces the first substring of this string that matches the given regular expression with the given replacement.
`String[]`	split(String regex) Splits this string around matches of the given regular expression.
`boolean`	startsWith(String prefix) Tests if this string starts with the specified prefix.
`boolean`	startsWith(String prefix, int toffset) Tests if the substring of this string beginning at the specified index starts with the specified prefix.
`String`	substring(int beginIndex, int endIndex) Returns a string that is a substring of this string.
`String`	toLowerCase() Converts all of the characters in this String to lower case using the rules of the default locale.
`String`	toString() This object (which is already a string!) is itself returned.
`String`	toUpperCase() Converts all of the characters in this String to upper case using the rules of the default locale.
`String`	trim() Returns a string whose value is this string, with any leading and trailing whitespace removed.
Note: Valid values for type are: boolean, double, float, int, long	
`static String`	valueOf(*type* b) Returns the string representation of the argument of type *type*.