Java

Sonstiges

```java
// Wrapper Klassen
// same for Integer, Double, Float, ...
Integer i = new Integer.valueOf(5);
Double d = new Double.valueOf("5");

Integer valueOf(String s);
Integer valueOf(int m);
double doubleValue();
float floatValue();
int intValue();
// for Integer wrapper class
int parseInt(String s);
// for Double wrapper
double parseDouble(String s);
```

```java
// Arrays
List l = Arrays.asList(new int[]{1, 2, 3});
```

JUnit

Annotations

```java
@BeforeClass
public static void beforeAllTests() {
    // Run once before ALL tests
}
```

```java
@Before
public void beforeEachTest() {
    // Run before each test
}
```

```java
@Test
public void test() {
    // Do the test
}
```

```java
@Test(expected = IllegalArgumentException.class)
public void testException() {
    // Do the test and expect an exception
}
```

```java
@Test(timeout = 250)
public void testTimeout() {
    // Do the test and time out after 250ms
}

@After
public void afterEachTest() {
    // Run after each test
}

@AfterClass
public static void afterAllTests() {
    // Run once after ALL tests
}
```

Methods

```java
// double, float
assertEquals(expected, actual, delta);
assertEquals(message, expected, actual, delta);
// int, long, objects
assertEquals(expected, actual);
assertEquals(message, expected, actual);
// same with ...
assertNotEquals(...);

// booleans
assertTrue(condition);
assertTrue(message, condition);
assertFalse(condition);
assertFalse(message, condition);

// null
assertNull(object);
assertNull(message, object);
// same with ...
assertNotNull(...);

// Arrays
// double, float
assertArrayEquals(expected, actual, delta);
assertArrayEquals(message, expected, actual, delta);
// all other
assertArrayEquals(expected, actual);
assertArrayEquals(message, expected, actual);
```

Strings

```
char charAt(int index);
int compareTo(Object o);
boolean contains(CharSequence str);
int compareTo(String str);
int compareToIgnoreCase(String str);
boolean equals(Object o);
boolean equalsIgnoreCase(String str);
int indexOf(String str);
int indexOf(String str, int fromIndex);
int lastIndexOf(String str);
int lastIndexOf(String str, int fromIndex);
int length();
boolean isEmpty();
String replace(char oldChar, char newChar);
boolean startsWith(String prefix);
boolean endsWith(String suffix);
String substring(int beginIndex);
String substring(int beginIndex, int endIndex);
char[] toCharArray();
String trim();
String toLowerCase();
String toUpperCase();
```

LinkedList

Constructor

```
LinkedList();
```

Methods

```
void add(int index, object element);
boolean add(object o);
boolean addAll(Collections c);
boolean addAll(int index, Collections c);
void clear();
Object clone();
boolean contains(Object o);
Object get(int index);
Object getFirst();
Object getLast();
int indexOf(Object o);
```

```
Object remove(int index);
Object remove(Object o);
Object set(int index, Object element);
int size();
Object[] toArray();
boolean contains(Object element);
Stream<T> stream();
```

ArrayList

Constructor

```
ArrayList();
ArrayList(Collection c);
ArrayList(int initialCapacity);
```

Methods

```
void ensureCapacity(int minCapacity)
void trimToSize()
```

HashMap

Constructor

```
HashMap();
HashMap(int initialCapacity);
HashMap(int initialCapacity, float loadFactor);
HashMap(Map m);
```

Methods

```
void clear();
Object clone();
boolean containsKey(Object key);
boolean containsValue(Object value);
Set entrySet();
Object get(Object key);
boolean isEmpty();
Set keySet();
Object put(Object key, Object value);
void putAll(Map m);
```

```java
Object remove(Object key);
int size();
Collection values();
```

Vector

Constructor

```java
Vector();
Vector(Collection c);
Vector(int initialCapacity);
Vector(int initialCapacity, int capacityIncrement);
```

Methods

```java
// Has the same methods as ArrayList
```

Streams

Erstellen von Streams

```java
default Stream<E> stream();
default Stream<E> parallelStream();
```

Intermediate operations (transformieren den Stream in einen neuen "Strom")

```java
Stream<T> filter(Predicate<? super T> predicate);
Stream<T> filter(p -> p.getAge() >= 15);
Stream<R> map(Function<? super T, ? super R> func);
Stream<R> map(e -> e * e);
// X = Double, Int, Long
Stream<E> mapToX(p -> p.getSize());
Stream<E> distinct();
Stream<T> sorted() / sorted(Comparator<? super T> comp);
Stream<T> limit(long maxSize);
Stream<T> skip(long n);
Stream<R> peek(Consumer<? super T> consumer);
Stream<R> peek(e -> System.out.println(e));
```

Terminal operations ("machen etwas" liefern aber keinen Stream)

```
void foreach(Consumer<? super T> action);
void foreach(e -> System.out.println(e));
long count();
Optional<T> min(Comparator<? super T> comp);
Optional<T> max(Comparator<? super T> comp);
Optional<T> max((person1, person2) -> Integer.compare(person1.getAge(), person2.getAge()));
boolean anyMatch(Predicate<? super T> pred);
boolean allMatch(Predicate<? super T> pred);
boolean noneMatch(Predicate<? super T> pred);
Object[] toArray();
List<X> collect(Collectors.toList());
```

Racket

Documentation

```
;; fcv: (listof number) number -> number
;; description
;; example: (fcv '(1 2 3 4 5) 3.6) -> 4
```

Tests

```
(check-expect expression expected-expression)
(check-expect (fahrenheit->celsius 212) 100)

(check-within expression expected-expression delta)
(check-within (roots-table (list 1.0 2.0 3.0))
                (list
                  (make-roots 1.0 1.0)
                  (make-roots 2  1.414)
                  (make-roots 3  1.713))
                0.1)

(check-error expression)
(check-error expression expected-error-message)
(check-error (lookup sample-table "kathi") "kathi not found")
```