interface Iterable\<T\>

Iterator\<\T\> iterator() Returns an iterator over elements of type \T\.
void forEach(Consumer\<? super \T\> action) Performs the given action for each element of this Iterable

public interface Collection\<E\> extends Iterable\<E\>

boolean add(E e) Ensures that this collection contains the specified element.
boolean addAll(Collection\<E\> c) Adds all of the elements in the specified collection to this collection.
void clear() Removes all of the elements from this collection.
boolean contains(Object o) Returns true if this collection contains the specified element.
boolean containsAll(Collection\<E\> c) Returns true if this collection contains all of the elements in the specified collection.
boolean isEmpty() Returns true if this collection contains no elements.
boolean removeAll(Collection\<E\> c) Removes a single instance of the specified element from this collection, if it is present.
boolean removeAll(Collection\<E\> c) Removes all of this collection’s elements that are also contained in the specified collection.
boolean removeIf(Predicate\<? super E\> filter) Removes all of the elements of this collection that satisfy the given predicate.
boolean retainAll(Collection\<E\> c) Retains only the elements in this collection that are contained in the specified collection.
int size() Returns the number of elements in this collection.
Stream\<E\> stream() Returns a sequential Stream with this collection as its source.

interface List\<E\> extends Collection\<E\>

void add(int index, E element) Inserts the specified element at the specified position in this list.
boolean addAll(int index, Collection\<E\> c) Inserts all of the elements in the specified collection into this list at the specified position.
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 it does not contain the element.
int lastIndexOf(Object o) Returns the index of the last occurrence of the specified element in this list, or -1 if it does not contain the element.
E remove(int index) Removes the element at the specified position in this list.
E set(int index, E element) Replaces the element at the specified position in this list with the specified element.
void sort(Comparator\<? super E\> c) Sorts this list according to the order induced by the specified Comparator.

interface Map\<K,V\>

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.
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\<K,V\> m) Copies all of the mappings from the specified map to this map.
V remove(Object key) Removes the mapping for a key from this map if it is present.
int size() Returns the number of key-value mappings in this map.

class String implements Comparable\<String\>

char charAt(int index) Returns the char value at the specified index.
boolean contains(String s) Returns true if and only if this string contains the specified string.
boolean endsWith(String suffix) Tests if this string ends with the specified suffix.
static String format(String format, Object... args) Returns a formatted string using the specified format string and arguments.
int indexOf(int ch) Returns the index of the first occurrence of the specified character.
int indexOf(int ch, int fromIndex) Returns the index of the first occurrence of the specified character, starting the search at the specified index.
int indexOf(String str) Returns the index of the first occurrence of the specified substring.
int indexOf(String str, int fromIndex) Returns the index of the first occurrence of the specified substring, starting at the specified index.
boolean isEmpty() Returns true if, and only if, length() is 0.
int lastIndexOf(int ch) Returns the index within this string of the last occurrence of the specified character.

int lastIndexOf(int ch, int fromIndex) Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.

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.

String replace(String target, String replacement) Replaces each substring of this string that matches the literal target string with the specified literal replacement string.

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.

String substring(int beginIndex) Returns a string that is a substring of this string.

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 toUpperCase() Converts all of the characters in this String to upper case using the rules of the default locale.

String trim() Converts any leading and trailing whitespace removed.

**class Scanner**

Scanner(InputStream source)

Constructs a new Scanner that produces values scanned from the specified input stream.

void close() Closes this scanner.

boolean hasNext() Returns true if this scanner has another token in its input.

boolean hasNextBoolean() Returns true if the next token in this scanner’s input can be interpreted as a boolean value using a case insensitive pattern created from the string "true | false".

boolean hasNextDouble() Returns true if the next token in this scanner’s input can be interpreted as a double using nextDouble().

boolean hasNextInt() Returns true if the next token in this scanner’s input can be interpreted as an int using nextInt().

boolean hasNextLine() Returns true if there is another line in the input of this scanner.

String next() Finds and returns the next complete token from this scanner.

Boolean nextBoolean() Scans the next token of the input as a boolean value and returns that value.

int nextInt() Scans the next token of the input as an int.

double nextDouble() Scans the next token of the input as a double.

**Functional interfaces**

Predicate<T> boolean test(T t) Evaluates this predicate on the given argument.

Supplier<T> T get() Gets a result.

Consumer<T> void accept(T t) Performs this operation on the given argument.

Function<T,R> R apply(T t) Applies this function to the given argument.

UnaryOperator<T> T apply(T t) Applies this function to the given argument.

BiFunction<T1,T2,R> R apply(T1 t1, T2 t2) Applies this function to the given arguments.

BinaryOperator<T> T apply(T t1, T t2) Applies this function to the given arguments.

**interface Stream<T>**

boolean allMatch(Predicate<? super T> predicate) Returns whether all elements of this stream match the provided predicate.

boolean anyMatch(Predicate<? super T> predicate) Returns whether any elements of this stream match the provided predicate.

<R,A> R collect(Collectors<? super T,A,R> collector) Performs a mutable reduction operation on the elements of this stream using a Collector.

Stream<T> filter(Predicate<? super T> predicate) Returns a stream consisting of the elements of this stream that match the given predicate.

void forEach(Consumer<? super T> action) Performs an action for each element of this stream.

Stream<R> map(Function<? super T,? extends R> mapper) Returns a stream consisting of the results of applying the given function to the elements of this stream.

T reduce(T identity, BinaryOperator<T> accumulator) Performs a reduction on the elements of this stream, using the provided identity value and an associative accumulation function, and returns the reduced value.
### Stream<T> sorted()

Returns a stream consisting of the elements of this stream, sorted according to natural order.

### Stream<T> sorted(Comparator<? super T> comparator)

Returns a stream consisting of the elements of this stream, sorted according to the provided Comparator.

---

**class Assert**

**protected Assert()**

Protect constructor since it is a static only class

Many of the methods below can be called with a variety of parameters.

**static void assertEquals(java.lang.Object expected, java.lang.Object actual)**

Asserts that two objects are equal.

**static void assertNotNull(java.lang.Object object)**

Asserts that an object isn’t null.

**static void assertNull(java.lang.Object object)**

Asserts that an object is null.

**static void assertTrue(boolean condition)**

Asserts that a condition is true.

**static void assertFalse(boolean condition)**

Asserts that a condition is false.

**static void fail()**

Fails a test with no message. With a String as a parameter it fails with the given string as message.