Modifier and Type	Field and Description
static boolean[]	**EMPTY_BOOLEAN_ARRAY** An empty immutable `boolean` array.
static **Boolean**[]	**EMPTY_BOOLEAN_OBJECT_ARRAY** An empty immutable **Boolean** array.
static byte[]	**EMPTY_BYTE_ARRAY** An empty immutable `byte` array.
static **Byte**[]	**EMPTY_BYTE_OBJECT_ARRAY** An empty immutable **Byte** array.
static char[]	**EMPTY_CHAR_ARRAY** An empty immutable `char` array.
static **Character**[]	**EMPTY_CHARACTER_OBJECT_ARRAY** An empty immutable **Character** array.
static **Class**<?>[]	**EMPTY_CLASS_ARRAY** An empty immutable **Class** array.
static double[]	**EMPTY_DOUBLE_ARRAY** An empty immutable `double` array.
static **Double**[]	**EMPTY_DOUBLE_OBJECT_ARRAY** An empty immutable **Double** array.
static **Field**[]	**EMPTY_FIELD_ARRAY** An empty immutable **Field** array.
static float[]	**EMPTY_FLOAT_ARRAY** An empty immutable `float` array.
static **Float**[]	**EMPTY_FLOAT_OBJECT_ARRAY** An empty immutable **Float** array.
static int[]	**EMPTY_INT_ARRAY** An empty immutable `int` array.
static **Integer**[]	**EMPTY_INTEGER_OBJECT_ARRAY** An empty immutable **Integer** array.
static long[]	**EMPTY_LONG_ARRAY** An empty immutable `long` array.
static **Long**[]	**EMPTY_LONG_OBJECT_ARRAY** An empty immutable **Long** array.

static **Method**[]	**EMPTY_METHOD_ARRAY** An empty immutable **Method** array.
static **Object**[]	**EMPTY_OBJECT_ARRAY** An empty immutable **Object** array.
static short[]	**EMPTY_SHORT_ARRAY** An empty immutable short array.
static **Short**[]	**EMPTY_SHORT_OBJECT_ARRAY** An empty immutable **Short** array.
static **String**[]	**EMPTY_STRING_ARRAY** An empty immutable **String** array.
static **Throwable**[]	**EMPTY_THROWABLE_ARRAY** An empty immutable **Throwable** array.
static **Type**[]	**EMPTY_TYPE_ARRAY** An empty immutable **Type** array.
static int	**INDEX_NOT_FOUND** The index value when an element is not found in a list or array: -1.

Modifier and Type	Method and Description
static boolean[]	**add**(boolean[] array, boolean element) Copies the given array and adds the given element at the end of the new array.
static boolean[]	**add**(boolean[] array, int index, boolean element) **Deprecated.** this method has been superseded by **insert(int, boolean[], boolean...)** and may be removed in a future release. Please note the handling of null input arrays differs in the new method: inserting X into a null array results in null not X.
static byte[]	**add**(byte[] array, byte element) Copies the given array and adds the given element at the end of the new array.

`static byte[]` **`add`**`(byte[] array, int index, byte element)`

Deprecated.

this method has been superseded by **`insert(int, byte[], byte...)`** and may be removed in a future release. Please note the handling of `null` input arrays differs in the new method: inserting `X` into a `null` array results in `null` not `X`.

`static char[]` **`add`**`(char[] array, char element)`

Copies the given array and adds the given element at the end of the new array.

`static char[]` **`add`**`(char[] array, int index, char element)`

Deprecated.

this method has been superseded by **`insert(int, char[], char...)`** and may be removed in a future release. Please note the handling of `null` input arrays differs in the new method: inserting `X` into a `null` array results in `null` not `X`.

`static double[]` **`add`**`(double[] array, double element)`

Copies the given array and adds the given element at the end of the new array.

`static double[]` **`add`**`(double[] array, int index, double element)`

Deprecated.

this method has been superseded by **`insert(int, double[], double...)`** and may be removed in a future release. Please note the handling of `null` input arrays differs in the new method: inserting `X` into a `null` array results in `null` not `X`.

`static float[]` **`add`**`(float[] array, float element)`

Copies the given array and adds the given element at the end of the new array.

`static float[]` **`add`**`(float[] array, int index, float element)`

Deprecated.

this method has been superseded by **`insert(int, float[], float...)`** and may be removed in a future release. Please note the handling of `null` input arrays differs in the new method: inserting `X` into a `null` array results in `null` not `X`.

`static int[]` **`add`**`(int[] array, int element)`

Copies the given array and adds the given element at the end of the new array.

static int[]	**add**(int[] array, int index, int element) **Deprecated.** this method has been superseded by **insert(int, int[], int...)** and may be removed in a future release. Please note the handling of `null` input arrays differs in the new method: inserting `X` into a `null` array results in `null` not `X`.
static long[]	**add**(long[] array, int index, long element) **Deprecated.** this method has been superseded by **insert(int, long[], long...)** and may be removed in a future release. Please note the handling of `null` input arrays differs in the new method: inserting `X` into a `null` array results in `null` not `X`.
static long[]	**add**(long[] array, long element) Copies the given array and adds the given element at the end of the new array.
static short[]	**add**(short[] array, int index, short element) **Deprecated.** this method has been superseded by **insert(int, short[], short...)** and may be removed in a future release. Please note the handling of `null` input arrays differs in the new method: inserting `X` into a `null` array results in `null` not `X`.
static short[]	**add**(short[] array, short element) Copies the given array and adds the given element at the end of the new array.
static <T> T[]	**add**(T[] array, int index, T element) **Deprecated.** this method has been superseded by **insert(int, T[], T...)** and may be removed in a future release. Please note the handling of `null` input arrays differs in the new method: inserting `X` into a `null` array results in `null` not `X`.
static <T> T[]	**add**(T[] array, T element) Copies the given array and adds the given element at the end of the new array.
static boolean[]	**addAll**(boolean[] array1, boolean... array2) Adds all the elements of the given arrays into a new array.
static byte[]	**addAll**(byte[] array1, byte... array2) Adds all the elements of the given arrays into a new array.
static char[]	**addAll**(char[] array1, char... array2) Adds all the elements of the given arrays into a new array.

`static double[]`	**`addAll`**`(double[] array1, double... array2)` Adds all the elements of the given arrays into a new array.
`static float[]`	**`addAll`**`(float[] array1, float... array2)` Adds all the elements of the given arrays into a new array.
`static int[]`	**`addAll`**`(int[] array1, int... array2)` Adds all the elements of the given arrays into a new array.
`static long[]`	**`addAll`**`(long[] array1, long... array2)` Adds all the elements of the given arrays into a new array.
`static short[]`	**`addAll`**`(short[] array1, short... array2)` Adds all the elements of the given arrays into a new array.
`static <T> T[]`	**`addAll`**`(T[] array1, T... array2)` Adds all the elements of the given arrays into a new array.
`static boolean[]`	**`addFirst`**`(boolean[] array, boolean element)` Copies the given array and adds the given element at the beginning of the new array.
`static byte[]`	**`addFirst`**`(byte[] array, byte element)` Copies the given array and adds the given element at the beginning of the new array.
`static char[]`	**`addFirst`**`(char[] array, char element)` Copies the given array and adds the given element at the beginning of the new array.
`static double[]`	**`addFirst`**`(double[] array, double element)` Copies the given array and adds the given element at the beginning of the new array.
`static float[]`	**`addFirst`**`(float[] array, float element)` Copies the given array and adds the given element at the beginning of the new array.
`static int[]`	**`addFirst`**`(int[] array, int element)` Copies the given array and adds the given element at the beginning of the new array.
`static long[]`	**`addFirst`**`(long[] array, long element)` Copies the given array and adds the given element at the beginning of the new array.
`static short[]`	**`addFirst`**`(short[] array, short element)` Copies the given array and adds the given element at the beginning of the new array.

static <T> T[]	**addFirst**(T[] array, T element) Copies the given array and adds the given element at the beginning of the new array.
static boolean[]	**clone**(boolean[] array) Clones an array returning a typecast result and handling `null`.
static byte[]	**clone**(byte[] array) Clones an array returning a typecast result and handling `null`.
static char[]	**clone**(char[] array) Clones an array returning a typecast result and handling `null`.
static double[]	**clone**(double[] array) Clones an array returning a typecast result and handling `null`.
static float[]	**clone**(float[] array) Clones an array returning a typecast result and handling `null`.
static int[]	**clone**(int[] array) Clones an array returning a typecast result and handling `null`.
static long[]	**clone**(long[] array) Clones an array returning a typecast result and handling `null`.
static short[]	**clone**(short[] array) Clones an array returning a typecast result and handling `null`.
static <T> T[]	**clone**(T[] array) Shallow clones an array returning a typecast result and handling `null`.
static boolean	**contains**(boolean[] array, boolean valueToFind) Checks if the value is in the given array.
static boolean	**contains**(byte[] array, byte valueToFind) Checks if the value is in the given array.
static boolean	**contains**(char[] array, char valueToFind) Checks if the value is in the given array.
static boolean	**contains**(double[] array, double valueToFind) Checks if the value is in the given array.
static boolean	**contains**(double[] array, double valueToFind, double tolerance) Checks if a value falling within the given tolerance is in the given array.

static boolean	**contains**(float[] array, float valueToFind) Checks if the value is in the given array.
static boolean	**contains**(int[] array, int valueToFind) Checks if the value is in the given array.
static boolean	**contains**(long[] array, long valueToFind) Checks if the value is in the given array.
static boolean	**contains**(**Object**[] array, **Object** objectToFind) Checks if the object is in the given array.
static boolean	**contains**(short[] array, short valueToFind) Checks if the value is in the given array.
static boolean	**containsAny**(**Object**[] array, **Object**... objectsToFind) Checks if any of the objects are in the given array.
static <T> T	**get**(T[] array, int index) Gets the nTh element of an array or null if the index is out of bounds or the array is null.
static <T> T	**get**(T[] array, int index, T defaultValue) Gets the nTh element of an array or a default value if the index is out of bounds.
static <T> **Class**<T>	**getComponentType**(T[] array) Gets an array's component type.
static int	**getLength**(**Object** array) Returns the length of the specified array.
static int	**hashCode**(**Object** array) Gets a hash code for an array handling multidimensional arrays correctly.
static **BitSet**	**indexesOf**(boolean[] array, boolean valueToFind) Finds the indices of the given value in the array.
static **BitSet**	**indexesOf**(boolean[] array, boolean valueToFind, int startIndex) Finds the indices of the given value in the array starting at the given index.
static **BitSet**	**indexesOf**(byte[] array, byte valueToFind) Finds the indices of the given value in the array.
static **BitSet**	**indexesOf**(byte[] array, byte valueToFind, int startIndex) Finds the indices of the given value in the array starting at the given index.

static **BitSet**	**indexesOf**(char[] array, char valueToFind) Finds the indices of the given value in the array.
static **BitSet**	**indexesOf**(char[] array, char valueToFind, int startIndex) Finds the indices of the given value in the array starting at the given index.
static **BitSet**	**indexesOf**(double[] array, double valueToFind) Finds the indices of the given value in the array.
static **BitSet**	**indexesOf**(double[] array, double valueToFind, double tolerance) Finds the indices of the given value within a given tolerance in the array.
static **BitSet**	**indexesOf**(double[] array, double valueToFind, int startIndex) Finds the indices of the given value in the array starting at the given index.
static **BitSet**	**indexesOf**(double[] array, double valueToFind, int startIndex, double tolerance) Finds the indices of the given value in the array starting at the given index.
static **BitSet**	**indexesOf**(float[] array, float valueToFind) Finds the indices of the given value in the array.
static **BitSet**	**indexesOf**(float[] array, float valueToFind, int startIndex) Finds the indices of the given value in the array starting at the given index.
static **BitSet**	**indexesOf**(int[] array, int valueToFind) Finds the indices of the given value in the array.
static **BitSet**	**indexesOf**(int[] array, int valueToFind, int startIndex) Finds the indices of the given value in the array starting at the given index.
static **BitSet**	**indexesOf**(long[] array, long valueToFind) Finds the indices of the given value in the array.
static **BitSet**	**indexesOf**(long[] array, long valueToFind, int startIndex) Finds the indices of the given value in the array starting at the given index.

static **BitSet**	**indexesOf**(**Object**[] array, **Object** objectToFind) Finds the indices of the given object in the array.
static **BitSet**	**indexesOf**(**Object**[] array, **Object** objectToFind, int startIndex) Finds the indices of the given object in the array starting at the given index.
static **BitSet**	**indexesOf**(short[] array, short valueToFind) Finds the indices of the given value in the array.
static **BitSet**	**indexesOf**(short[] array, short valueToFind, int startIndex) Finds the indices of the given value in the array starting at the given index.
static int	**indexOf**(boolean[] array, boolean valueToFind) Finds the index of the given value in the array.
static int	**indexOf**(boolean[] array, boolean valueToFind, int startIndex) Finds the index of the given value in the array starting at the given index.
static int	**indexOf**(byte[] array, byte valueToFind) Finds the index of the given value in the array.
static int	**indexOf**(byte[] array, byte valueToFind, int startIndex) Finds the index of the given value in the array starting at the given index.
static int	**indexOf**(char[] array, char valueToFind) Finds the index of the given value in the array.
static int	**indexOf**(char[] array, char valueToFind, int startIndex) Finds the index of the given value in the array starting at the given index.
static int	**indexOf**(double[] array, double valueToFind) Finds the index of the given value in the array.
static int	**indexOf**(double[] array, double valueToFind, double tolerance) Finds the index of the given value within a given tolerance in the array.
static int	**indexOf**(double[] array, double valueToFind, int startIndex)

Finds the index of the given value in the array starting at the given index.

static int	**indexOf**(double[] array, double valueToFind, int startIndex, double tolerance) Finds the index of the given value in the array starting at the given index.
static int	**indexOf**(float[] array, float valueToFind) Finds the index of the given value in the array.
static int	**indexOf**(float[] array, float valueToFind, int startIndex) Finds the index of the given value in the array starting at the given index.
static int	**indexOf**(int[] array, int valueToFind) Finds the index of the given value in the array.
static int	**indexOf**(int[] array, int valueToFind, int startIndex) Finds the index of the given value in the array starting at the given index.
static int	**indexOf**(long[] array, long valueToFind) Finds the index of the given value in the array.
static int	**indexOf**(long[] array, long valueToFind, int startIndex) Finds the index of the given value in the array starting at the given index.
static int	**indexOf**(**Object**[] array, **Object** objectToFind) Finds the index of the given object in the array.
static int	**indexOf**(**Object**[] array, **Object** objectToFind, int startIndex) Finds the index of the given object in the array starting at the given index.
static int	**indexOf**(short[] array, short valueToFind) Finds the index of the given value in the array.
static int	**indexOf**(short[] array, short valueToFind, int startIndex) Finds the index of the given value in the array starting at the given index.
static boolean[]	**insert**(int index, boolean[] array, boolean... values) Inserts elements into an array at the given index (starting from zero).

`static byte[]`	**`insert`**`(int index, byte[] array, byte... values)` Inserts elements into an array at the given index (starting from zero).
`static char[]`	**`insert`**`(int index, char[] array, char... values)` Inserts elements into an array at the given index (starting from zero).
`static double[]`	**`insert`**`(int index, double[] array, double... values)` Inserts elements into an array at the given index (starting from zero).
`static float[]`	**`insert`**`(int index, float[] array, float... values)` Inserts elements into an array at the given index (starting from zero).
`static int[]`	**`insert`**`(int index, int[] array, int... values)` Inserts elements into an array at the given index (starting from zero).
`static long[]`	**`insert`**`(int index, long[] array, long... values)` Inserts elements into an array at the given index (starting from zero).
`static short[]`	**`insert`**`(int index, short[] array, short... values)` Inserts elements into an array at the given index (starting from zero).
`static <T> T[]`	**`insert`**`(int index, T[] array, T... values)` Inserts elements into an array at the given index (starting from zero).
`static <T> boolean`	**`isArrayIndexValid`**`(T[] array, int index)` Returns whether a given array can safely be accessed at the given index.
`static boolean`	**`isEmpty`**`(boolean[] array)` Checks if an array of primitive booleans is empty or `null`.
`static boolean`	**`isEmpty`**`(byte[] array)` Checks if an array of primitive bytes is empty or `null`.
`static boolean`	**`isEmpty`**`(char[] array)` Checks if an array of primitive chars is empty or `null`.
`static boolean`	**`isEmpty`**`(double[] array)` Checks if an array of primitive doubles is empty or `null`.
`static boolean`	**`isEmpty`**`(float[] array)` Checks if an array of primitive floats is empty or `null`.
`static boolean`	**`isEmpty`**`(int[] array)` Checks if an array of primitive ints is empty or `null`.
`static boolean`	**`isEmpty`**`(long[] array)` Checks if an array of primitive longs is empty or `null`.

static boolean	**isEmpty**(**Object**[] array) Checks if an array of Objects is empty or `null`.
static boolean	**isEmpty**(short[] array) Checks if an array of primitive shorts is empty or `null`.
static boolean	**isEquals**(**Object** array1, **Object** array2) **Deprecated.** this method has been replaced by `java.util.Objects.deepEquals(Object, Object)` and will be removed from future releases.
static boolean	**isNotEmpty**(boolean[] array) Checks if an array of primitive booleans is not empty and not `null`.
static boolean	**isNotEmpty**(byte[] array) Checks if an array of primitive bytes is not empty and not `null`.
static boolean	**isNotEmpty**(char[] array) Checks if an array of primitive chars is not empty and not `null`.
static boolean	**isNotEmpty**(double[] array) Checks if an array of primitive doubles is not empty and not `null`.
static boolean	**isNotEmpty**(float[] array) Checks if an array of primitive floats is not empty and not `null`.
static boolean	**isNotEmpty**(int[] array) Checks if an array of primitive ints is not empty and not `null`.
static boolean	**isNotEmpty**(long[] array) Checks if an array of primitive longs is not empty and not `null`.
static boolean	**isNotEmpty**(short[] array) Checks if an array of primitive shorts is not empty and not `null`.
static <T> boolean	**isNotEmpty**(T[] array) Checks if an array of Objects is not empty and not `null`.
static boolean	**isSameLength**(boolean[] array1, boolean[] array2) Checks whether two arrays are the same length, treating `null` arrays as length `0`.
static boolean	**isSameLength**(byte[] array1, byte[] array2) Checks whether two arrays are the same length, treating `null` arrays as length `0`.
static boolean	**isSameLength**(char[] array1, char[] array2)

Checks whether two arrays are the same length, treating `null` arrays as length `0`.

`static boolean` **`isSameLength`**`(double[] array1, double[] array2)`

Checks whether two arrays are the same length, treating `null` arrays as length `0`.

`static boolean` **`isSameLength`**`(float[] array1, float[] array2)`

Checks whether two arrays are the same length, treating `null` arrays as length `0`.

`static boolean` **`isSameLength`**`(int[] array1, int[] array2)`

Checks whether two arrays are the same length, treating `null` arrays as length `0`.

`static boolean` **`isSameLength`**`(long[] array1, long[] array2)`

Checks whether two arrays are the same length, treating `null` arrays as length `0`.

`static boolean` **`isSameLength`**`(Object[] array1, Object[] array2)`

Checks whether two arrays are the same length, treating `null` arrays as length `0`.

`static boolean` **`isSameLength`**`(Object array1, Object array2)`

Checks whether two arrays are the same length, treating `null` arrays as length `0`.

`static boolean` **`isSameLength`**`(short[] array1, short[] array2)`

Checks whether two arrays are the same length, treating `null` arrays as length `0`.

`static boolean` **`isSameType`**`(Object array1, Object array2)`

Checks whether two arrays are the same type taking into account multidimensional arrays.

`static boolean` **`isSorted`**`(boolean[] array)`

This method checks whether the provided array is sorted according to natural ordering (`false` before `true`).

`static boolean` **`isSorted`**`(byte[] array)`

Checks whether the provided array is sorted according to natural ordering.

`static boolean` **`isSorted`**`(char[] array)`

Checks whether the provided array is sorted according to natural ordering.

`static boolean` **`isSorted`**`(double[] array)`

This method checks whether the provided array is sorted according to natural ordering.

static boolean	**isSorted**(float[] array) This method checks whether the provided array is sorted according to natural ordering.
static boolean	**isSorted**(int[] array) This method checks whether the provided array is sorted according to natural ordering.
static boolean	**isSorted**(long[] array) This method checks whether the provided array is sorted according to natural ordering.
static boolean	**isSorted**(short[] array) This method checks whether the provided array is sorted according to natural ordering.
static <T extends **Comparable**<? super T>> boolean	**isSorted**(T[] array) This method checks whether the provided array is sorted according to the class's `compareTo` method.
static <T> boolean	**isSorted**(T[] array, **Comparator**<T> comparator) This method checks whether the provided array is sorted according to the provided **Comparator**.
static int	**lastIndexOf**(boolean[] array, boolean valueToFind) Finds the last index of the given value within the array.
static int	**lastIndexOf**(boolean[] array, boolean valueToFind, int startIndex) Finds the last index of the given value in the array starting at the given index.
static int	**lastIndexOf**(byte[] array, byte valueToFind) Finds the last index of the given value within the array.
static int	**lastIndexOf**(byte[] array, byte valueToFind, int startIndex) Finds the last index of the given value in the array starting at the given index.
static int	**lastIndexOf**(char[] array, char valueToFind) Finds the last index of the given value within the array.
static int	**lastIndexOf**(char[] array, char valueToFind, int startIndex) Finds the last index of the given value in the array starting at the given index.

static int	**lastIndexOf**(double[] array, double valueToFind) Finds the last index of the given value within the array.
static int	**lastIndexOf**(double[] array, double valueToFind, double tolerance) Finds the last index of the given value within a given tolerance in the array.
static int	**lastIndexOf**(double[] array, double valueToFind, int startIndex) Finds the last index of the given value in the array starting at the given index.
static int	**lastIndexOf**(double[] array, double valueToFind, int startIndex, double tolerance) Finds the last index of the given value in the array starting at the given index.
static int	**lastIndexOf**(float[] array, float valueToFind) Finds the last index of the given value within the array.
static int	**lastIndexOf**(float[] array, float valueToFind, int startIndex) Finds the last index of the given value in the array starting at the given index.
static int	**lastIndexOf**(int[] array, int valueToFind) Finds the last index of the given value within the array.
static int	**lastIndexOf**(int[] array, int valueToFind, int startIndex) Finds the last index of the given value in the array starting at the given index.
static int	**lastIndexOf**(long[] array, long valueToFind) Finds the last index of the given value within the array.
static int	**lastIndexOf**(long[] array, long valueToFind, int startIndex) Finds the last index of the given value in the array starting at the given index.
static int	**lastIndexOf**(**Object**[] array, **Object** objectToFind) Finds the last index of the given object within the array.
static int	**lastIndexOf**(**Object**[] array, **Object** objectToFind, int startIndex) Finds the last index of the given object in the array starting at the given index.

static int	**lastIndexOf**(short[] array, short valueToFind) Finds the last index of the given value within the array.
static int	**lastIndexOf**(short[] array, short valueToFind, int startIndex) Finds the last index of the given value in the array starting at the given index.
static <T> T[]	**newInstance**(**Class**<T> componentType, int length) Delegates to **Array.newInstance(Class,int)** using generics.
static boolean[]	**nullToEmpty**(boolean[] array) Defensive programming technique to change a null reference to an empty one.
static **Boolean**[]	**nullToEmpty**(**Boolean**[] array) Defensive programming technique to change a null reference to an empty one.
static byte[]	**nullToEmpty**(byte[] array) Defensive programming technique to change a null reference to an empty one.
static **Byte**[]	**nullToEmpty**(**Byte**[] array) Defensive programming technique to change a null reference to an empty one.
static char[]	**nullToEmpty**(char[] array) Defensive programming technique to change a null reference to an empty one.
static **Character**[]	**nullToEmpty**(**Character**[] array) Defensive programming technique to change a null reference to an empty one.
static **Class**<?>[]	**nullToEmpty**(**Class**<?>[] array) Defensive programming technique to change a null reference to an empty one.
static double[]	**nullToEmpty**(double[] array) Defensive programming technique to change a null reference to an empty one.
static **Double**[]	**nullToEmpty**(**Double**[] array) Defensive programming technique to change a null reference to an empty one.
static float[]	**nullToEmpty**(float[] array) Defensive programming technique to change a null reference to an empty one.

static **Float**[]	**nullToEmpty**(**Float**[] array) Defensive programming technique to change a null reference to an empty one.
static int[]	**nullToEmpty**(int[] array) Defensive programming technique to change a null reference to an empty one.
static **Integer**[]	**nullToEmpty**(**Integer**[] array) Defensive programming technique to change a null reference to an empty one.
static long[]	**nullToEmpty**(long[] array) Defensive programming technique to change a null reference to an empty one.
static **Long**[]	**nullToEmpty**(**Long**[] array) Defensive programming technique to change a null reference to an empty one.
static **Object**[]	**nullToEmpty**(**Object**[] array) Defensive programming technique to change a null reference to an empty one.
static short[]	**nullToEmpty**(short[] array) Defensive programming technique to change a null reference to an empty one.
static **Short**[]	**nullToEmpty**(**Short**[] array) Defensive programming technique to change a null reference to an empty one.
static **String**[]	**nullToEmpty**(**String**[] array) Defensive programming technique to change a null reference to an empty one.
static <T> T[]	**nullToEmpty**(T[] array, **Class**<T[]> type) Defensive programming technique to change a null reference to an empty one.
static boolean[]	**remove**(boolean[] array, int index) Removes the element at the specified position from the specified array.
static byte[]	**remove**(byte[] array, int index) Removes the element at the specified position from the specified array.
static char[]	**remove**(char[] array, int index)

Removes the element at the specified position from the specified array.

`static double[]` **`remove`**`(double[] array, int index)`
Removes the element at the specified position from the specified array.

`static float[]` **`remove`**`(float[] array, int index)`
Removes the element at the specified position from the specified array.

`static int[]` **`remove`**`(int[] array, int index)`
Removes the element at the specified position from the specified array.

`static long[]` **`remove`**`(long[] array, int index)`
Removes the element at the specified position from the specified array.

`static short[]` **`remove`**`(short[] array, int index)`
Removes the element at the specified position from the specified array.

`static <T> T[]` **`remove`**`(T[] array, int index)`
Removes the element at the specified position from the specified array.

`static boolean[]` **`removeAll`**`(boolean[] array, int... indices)`
Removes the elements at the specified positions from the specified array.

`static byte[]` **`removeAll`**`(byte[] array, int... indices)`
Removes the elements at the specified positions from the specified array.

`static char[]` **`removeAll`**`(char[] array, int... indices)`
Removes the elements at the specified positions from the specified array.

`static double[]` **`removeAll`**`(double[] array, int... indices)`
Removes the elements at the specified positions from the specified array.

`static float[]` **`removeAll`**`(float[] array, int... indices)`
Removes the elements at the specified positions from the specified array.

`static int[]` **`removeAll`**`(int[] array, int... indices)`
Removes the elements at the specified positions from the specified array.

static long[]	**removeAll**(long[] array, int... indices) Removes the elements at the specified positions from the specified array.
static short[]	**removeAll**(short[] array, int... indices) Removes the elements at the specified positions from the specified array.
static <T> T[]	**removeAll**(T[] array, int... indices) Removes the elements at the specified positions from the specified array.
static boolean[]	**removeAllOccurences**(boolean[] array, boolean element) **Deprecated.** Use **removeAllOccurrences(boolean[], boolean)**
static byte[]	**removeAllOccurences**(byte[] array, byte element) **Deprecated.** Use **removeAllOccurrences(byte[], byte)**
static char[]	**removeAllOccurences**(char[] array, char element) **Deprecated.** Use **removeAllOccurrences(char[], char)**
static double[]	**removeAllOccurences**(double[] array, double element) **Deprecated.** Use **removeAllOccurrences(double[], double)**
static float[]	**removeAllOccurences**(float[] array, float element) **Deprecated.** Use **removeAllOccurrences(float[], float)**
static int[]	**removeAllOccurences**(int[] array, int element) **Deprecated.** Use **removeAllOccurrences(int[], int)**
static long[]	**removeAllOccurences**(long[] array, long element) **Deprecated.** Use **removeAllOccurrences(long[], long)**
static short[]	**removeAllOccurences**(short[] array, short element) **Deprecated.** Use **removeAllOccurrences(short[], short)**

static <T> T[]	**removeAllOccurences**(T[] array, T element) **Deprecated.** Use **removeAllOccurrences(Object[], Object)**
static boolean[]	**removeAllOccurrences**(boolean[] array, boolean element) Removes the occurrences of the specified element from the specified boolean array.
static byte[]	**removeAllOccurrences**(byte[] array, byte element) Removes the occurrences of the specified element from the specified byte array.
static char[]	**removeAllOccurrences**(char[] array, char element) Removes the occurrences of the specified element from the specified char array.
static double[]	**removeAllOccurrences**(double[] array, double element) Removes the occurrences of the specified element from the specified double array.
static float[]	**removeAllOccurrences**(float[] array, float element) Removes the occurrences of the specified element from the specified float array.
static int[]	**removeAllOccurrences**(int[] array, int element) Removes the occurrences of the specified element from the specified int array.
static long[]	**removeAllOccurrences**(long[] array, long element) Removes the occurrences of the specified element from the specified long array.
static short[]	**removeAllOccurrences**(short[] array, short element) Removes the occurrences of the specified element from the specified short array.
static <T> T[]	**removeAllOccurrences**(T[] array, T element) Removes the occurrences of the specified element from the specified array.
static boolean[]	**removeElement**(boolean[] array, boolean element) Removes the first occurrence of the specified element from the specified array.
static byte[]	**removeElement**(byte[] array, byte element) Removes the first occurrence of the specified element from the specified array.

static char[]	**removeElement**(char[] array, char element) Removes the first occurrence of the specified element from the specified array.
static double[]	**removeElement**(double[] array, double element) Removes the first occurrence of the specified element from the specified array.
static float[]	**removeElement**(float[] array, float element) Removes the first occurrence of the specified element from the specified array.
static int[]	**removeElement**(int[] array, int element) Removes the first occurrence of the specified element from the specified array.
static long[]	**removeElement**(long[] array, long element) Removes the first occurrence of the specified element from the specified array.
static short[]	**removeElement**(short[] array, short element) Removes the first occurrence of the specified element from the specified array.
static <T> T[]	**removeElement**(T[] array, **Object** element) Removes the first occurrence of the specified element from the specified array.
static boolean[]	**removeElements**(boolean[] array, boolean... values) Removes occurrences of specified elements, in specified quantities, from the specified array.
static byte[]	**removeElements**(byte[] array, byte... values) Removes occurrences of specified elements, in specified quantities, from the specified array.
static char[]	**removeElements**(char[] array, char... values) Removes occurrences of specified elements, in specified quantities, from the specified array.
static double[]	**removeElements**(double[] array, double... values) Removes occurrences of specified elements, in specified quantities, from the specified array.
static float[]	**removeElements**(float[] array, float... values) Removes occurrences of specified elements, in specified quantities, from the specified array.
static int[]	**removeElements**(int[] array, int... values) Removes occurrences of specified elements, in specified quantities, from the specified array.

static long[]	**removeElements**(long[] array, long... values) Removes occurrences of specified elements, in specified quantities, from the specified array.
static short[]	**removeElements**(short[] array, short... values) Removes occurrences of specified elements, in specified quantities, from the specified array.
static <T> T[]	**removeElements**(T[] array, T... values) Removes occurrences of specified elements, in specified quantities, from the specified array.
static void	**reverse**(boolean[] array) Reverses the order of the given array.
static void	**reverse**(boolean[] array, int startIndexInclusive, int endIndexExclusive) Reverses the order of the given array in the given range.
static void	**reverse**(byte[] array) Reverses the order of the given array.
static void	**reverse**(byte[] array, int startIndexInclusive, int endIndexExclusive) Reverses the order of the given array in the given range.
static void	**reverse**(char[] array) Reverses the order of the given array.
static void	**reverse**(char[] array, int startIndexInclusive, int endIndexExclusive) Reverses the order of the given array in the given range.
static void	**reverse**(double[] array) Reverses the order of the given array.
static void	**reverse**(double[] array, int startIndexInclusive, int endIndexExclusive) Reverses the order of the given array in the given range.
static void	**reverse**(float[] array) Reverses the order of the given array.
static void	**reverse**(float[] array, int startIndexInclusive, int endIndexExclusive) Reverses the order of the given array in the given range.
static void	**reverse**(int[] array) Reverses the order of the given array.

static void	**reverse**(int[] array, int startIndexInclusive, int endIndexExclusive) Reverses the order of the given array in the given range.
static void	**reverse**(long[] array) Reverses the order of the given array.
static void	**reverse**(long[] array, int startIndexInclusive, int endIndexExclusive) Reverses the order of the given array in the given range.
static void	**reverse**(**Object**[] array) Reverses the order of the given array.
static void	**reverse**(**Object**[] array, int startIndexInclusive, int endIndexExclusive) Reverses the order of the given array in the given range.
static void	**reverse**(short[] array) Reverses the order of the given array.
static void	**reverse**(short[] array, int startIndexInclusive, int endIndexExclusive) Reverses the order of the given array in the given range.
static <T> T[]	**setAll**(T[] array, **IntFunction**<? extends T> generator) Sets all elements of the specified array, using the provided generator supplier to compute each element.
static <T> T[]	**setAll**(T[] array, **Supplier**<? extends T> generator) Sets all elements of the specified array, using the provided generator supplier to compute each element.
static void	**shift**(boolean[] array, int offset) Shifts the order of the given boolean array.
static void	**shift**(boolean[] array, int startIndexInclusive, int endIndexExclusive, int offset) Shifts the order of a series of elements in the given boolean array.
static void	**shift**(byte[] array, int offset) Shifts the order of the given byte array.
static void	**shift**(byte[] array, int startIndexInclusive, int endIndexExclusive, int offset) Shifts the order of a series of elements in the given byte array.
static void	**shift**(char[] array, int offset) Shifts the order of the given char array.

static void	**shift**(char[] array, int startIndexInclusive, int endIndexExclusive, int offset) Shifts the order of a series of elements in the given char array.
static void	**shift**(double[] array, int offset) Shifts the order of the given double array.
static void	**shift**(double[] array, int startIndexInclusive, int endIndexExclusive, int offset) Shifts the order of a series of elements in the given double array.
static void	**shift**(float[] array, int offset) Shifts the order of the given float array.
static void	**shift**(float[] array, int startIndexInclusive, int endIndexExclusive, int offset) Shifts the order of a series of elements in the given float array.
static void	**shift**(int[] array, int offset) Shifts the order of the given int array.
static void	**shift**(int[] array, int startIndexInclusive, int endIndexExclusive, int offset) Shifts the order of a series of elements in the given int array.
static void	**shift**(long[] array, int offset) Shifts the order of the given long array.
static void	**shift**(long[] array, int startIndexInclusive, int endIndexExclusive, int offset) Shifts the order of a series of elements in the given long array.
static void	**shift**(**Object**[] array, int offset) Shifts the order of the given array.
static void	**shift**(**Object**[] array, int startIndexInclusive, int endIndexExclusive, int offset) Shifts the order of a series of elements in the given array.
static void	**shift**(short[] array, int offset) Shifts the order of the given short array.
static void	**shift**(short[] array, int startIndexInclusive, int endIndexExclusive, int offset) Shifts the order of a series of elements in the given short array.
static void	**shuffle**(boolean[] array) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.

static void	**shuffle**(boolean[] array, **Random** random) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static void	**shuffle**(byte[] array) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static void	**shuffle**(byte[] array, **Random** random) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static void	**shuffle**(char[] array) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static void	**shuffle**(char[] array, **Random** random) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static void	**shuffle**(double[] array) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static void	**shuffle**(double[] array, **Random** random) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static void	**shuffle**(float[] array) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static void	**shuffle**(float[] array, **Random** random) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static void	**shuffle**(int[] array) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static void	**shuffle**(int[] array, **Random** random) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static void	**shuffle**(long[] array) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static void	**shuffle**(long[] array, **Random** random) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.

static void	**shuffle**(**Object**[] array) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static void	**shuffle**(**Object**[] array, **Random** random) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static void	**shuffle**(short[] array) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static void	**shuffle**(short[] array, **Random** random) Randomly permutes the elements of the specified array using the Fisher-Yates algorithm.
static boolean[]	**subarray**(boolean[] array, int startIndexInclusive, int endIndexExclusive) Produces a new `boolean` array containing the elements between the start and end indices.
static byte[]	**subarray**(byte[] array, int startIndexInclusive, int endIndexExclusive) Produces a new `byte` array containing the elements between the start and end indices.
static char[]	**subarray**(char[] array, int startIndexInclusive, int endIndexExclusive) Produces a new `char` array containing the elements between the start and end indices.
static double[]	**subarray**(double[] array, int startIndexInclusive, int endIndexExclusive) Produces a new `double` array containing the elements between the start and end indices.
static float[]	**subarray**(float[] array, int startIndexInclusive, int endIndexExclusive) Produces a new `float` array containing the elements between the start and end indices.
static int[]	**subarray**(int[] array, int startIndexInclusive, int endIndexExclusive) Produces a new `int` array containing the elements between the start and end indices.
static long[]	**subarray**(long[] array, int startIndexInclusive, int endIndexExclusive) Produces a new `long` array containing the elements between the start and end indices.

static short[]	**subarray**(short[] array, int startIndexInclusive, int endIndexExclusive) Produces a new short array containing the elements between the start and end indices.
static <T> T[]	**subarray**(T[] array, int startIndexInclusive, int endIndexExclusive) Produces a new array containing the elements between the start and end indices.
static void	**swap**(boolean[] array, int offset1, int offset2) Swaps two elements in the given boolean array.
static void	**swap**(boolean[] array, int offset1, int offset2, int len) Swaps a series of elements in the given boolean array.
static void	**swap**(byte[] array, int offset1, int offset2) Swaps two elements in the given byte array.
static void	**swap**(byte[] array, int offset1, int offset2, int len) Swaps a series of elements in the given byte array.
static void	**swap**(char[] array, int offset1, int offset2) Swaps two elements in the given char array.
static void	**swap**(char[] array, int offset1, int offset2, int len) Swaps a series of elements in the given char array.
static void	**swap**(double[] array, int offset1, int offset2) Swaps two elements in the given double array.
static void	**swap**(double[] array, int offset1, int offset2, int len) Swaps a series of elements in the given double array.
static void	**swap**(float[] array, int offset1, int offset2) Swaps two elements in the given float array.
static void	**swap**(float[] array, int offset1, int offset2, int len) Swaps a series of elements in the given float array.
static void	**swap**(int[] array, int offset1, int offset2) Swaps two elements in the given int array.
static void	**swap**(int[] array, int offset1, int offset2, int len) Swaps a series of elements in the given int array.

static void	**swap**(long[] array, int offset1, int offset2) Swaps two elements in the given long array.
static void	**swap**(long[] array, int offset1, int offset2, int len) Swaps a series of elements in the given long array.
static void	**swap**(**Object**[] array, int offset1, int offset2) Swaps two elements in the given array.
static void	**swap**(**Object**[] array, int offset1, int offset2, int len) Swaps a series of elements in the given array.
static void	**swap**(short[] array, int offset1, int offset2) Swaps two elements in the given short array.
static void	**swap**(short[] array, int offset1, int offset2, int len) Swaps a series of elements in the given short array.
static <T> T[]	**toArray**(T... items) Create a type-safe generic array.
static **Map**<**Object**,**Object**>	**toMap**(**Object**[] array) Converts the given array into a **Map**.
static **Boolean**[]	**toObject**(boolean[] array) Converts an array of primitive booleans to objects.
static **Byte**[]	**toObject**(byte[] array) Converts an array of primitive bytes to objects.
static **Character**[]	**toObject**(char[] array) Converts an array of primitive chars to objects.
static **Double**[]	**toObject**(double[] array) Converts an array of primitive doubles to objects.
static **Float**[]	**toObject**(float[] array) Converts an array of primitive floats to objects.
static **Integer**[]	**toObject**(int[] array) Converts an array of primitive ints to objects.
static **Long**[]	**toObject**(long[] array) Converts an array of primitive longs to objects.
static **Short**[]	**toObject**(short[] array) Converts an array of primitive shorts to objects.

static boolean[]	**toPrimitive**(**Boolean**[] array) Converts an array of object Booleans to primitives.
static boolean[]	**toPrimitive**(**Boolean**[] array, boolean valueForNull) Converts an array of object Booleans to primitives handling `null`.
static byte[]	**toPrimitive**(**Byte**[] array) Converts an array of object Bytes to primitives.
static byte[]	**toPrimitive**(**Byte**[] array, byte valueForNull) Converts an array of object Bytes to primitives handling `null`.
static char[]	**toPrimitive**(**Character**[] array) Converts an array of object Characters to primitives.
static char[]	**toPrimitive**(**Character**[] array, char valueForNull) Converts an array of object Character to primitives handling `null`.
static double[]	**toPrimitive**(**Double**[] array) Converts an array of object Doubles to primitives.
static double[]	**toPrimitive**(**Double**[] array, double valueForNull) Converts an array of object Doubles to primitives handling `null`.
static float[]	**toPrimitive**(**Float**[] array) Converts an array of object Floats to primitives.
static float[]	**toPrimitive**(**Float**[] array, float valueForNull) Converts an array of object Floats to primitives handling `null`.
static int[]	**toPrimitive**(**Integer**[] array) Converts an array of object Integers to primitives.
static int[]	**toPrimitive**(**Integer**[] array, int valueForNull) Converts an array of object Integer to primitives handling `null`.
static long[]	**toPrimitive**(**Long**[] array) Converts an array of object Longs to primitives.
static long[]	**toPrimitive**(**Long**[] array, long valueForNull) Converts an array of object Long to primitives handling `null`.
static **Object**	**toPrimitive**(**Object** array) Create an array of primitive type from an array of wrapper types.
static short[]	**toPrimitive**(**Short**[] array) Converts an array of object Shorts to primitives.
static short[]	**toPrimitive**(**Short**[] array, short valueForNull) Converts an array of object Short to primitives handling `null`.

static **String**	**toString**(**Object** array) Outputs an array as a String, treating `null` as an empty array.
static **String**	**toString**(**Object** array, **String** stringIfNull) Outputs an array as a String handling `null`s.
static **String**[]	**toStringArray**(**Object**[] array) Returns an array containing the string representation of each element in the argument array.
static **String**[]	**toStringArray**(**Object**[] array, **String** valueForNullElements) Returns an array containing the string representation of each element in the argument array handling `null` elements.

Methods inherited from class java.lang.Object

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`