`static <T>` **`List`**`<T>`	**`asList`**`(T... a)` Returns a fixed-size list backed by the specified array.
`static int`	**`binarySearch`**`(byte[] a, byte key)` Searches the specified array of bytes for the specified value using the binary search algorithm.
`static int`	**`binarySearch`**`(byte[] a, int fromIndex, int toIndex, byte key)` Searches a range of the specified array of bytes for the specified value using the binary search algorithm.
`static int`	**`binarySearch`**`(char[] a, char key)` Searches the specified array of chars for the specified value using the binary search algorithm.
`static int`	**`binarySearch`**`(char[] a, int fromIndex, int toIndex, char key)` Searches a range of the specified array of chars for the specified value using the binary search algorithm.
`static int`	**`binarySearch`**`(double[] a, double key)` Searches the specified array of doubles for the specified value using the binary search algorithm.
`static int`	**`binarySearch`**`(double[] a, int fromIndex, int toIndex, double key)` Searches a range of the specified array of doubles for the specified value using the binary search algorithm.
`static int`	**`binarySearch`**`(float[] a, float key)` Searches the specified array of floats for the specified value using the binary search algorithm.
`static int`	**`binarySearch`**`(float[] a, int fromIndex, int toIndex, float key)` Searches a range of the specified array of floats for the specified value using the binary search algorithm.
`static int`	**`binarySearch`**`(int[] a, int key)` Searches the specified array of ints for the specified value using the binary search algorithm.
`static int`	**`binarySearch`**`(int[] a, int fromIndex, int toIndex, int key)` Searches a range of the specified array of ints for the specified value using the binary search algorithm.
`static int`	**`binarySearch`**`(long[] a, int fromIndex, int toIndex, long key)`

	Searches a range of the specified array of longs for the specified value using the binary search algorithm.
`static int`	**`binarySearch`**`(long[] a, long key)`
	Searches the specified array of longs for the specified value using the binary search algorithm.
`static int`	**`binarySearch`**`(`**`Object`**`[] a, int fromIndex, int toIndex, `**`Object`**` key)`
	Searches a range of the specified array for the specified object using the binary search algorithm.
`static int`	**`binarySearch`**`(`**`Object`**`[] a, `**`Object`**` key)`
	Searches the specified array for the specified object using the binary search algorithm.
`static int`	**`binarySearch`**`(short[] a, int fromIndex, int toIndex, short key)`
	Searches a range of the specified array of shorts for the specified value using the binary search algorithm.
`static int`	**`binarySearch`**`(short[] a, short key)`
	Searches the specified array of shorts for the specified value using the binary search algorithm.
`static <T> int`	**`binarySearch`**`(T[] a, int fromIndex, int toIndex, T key, `**`Comparator`**`<? super T> c)`
	Searches a range of the specified array for the specified object using the binary search algorithm.
`static <T> int`	**`binarySearch`**`(T[] a, T key, `**`Comparator`**`<? super T> c)`
	Searches the specified array for the specified object using the binary search algorithm.
`static boolean[]`	**`copyOf`**`(boolean[] original, int newLength)`
	Copies the specified array, truncating or padding with `false` (if necessary) so the copy has the specified length.
`static byte[]`	**`copyOf`**`(byte[] original, int newLength)`
	Copies the specified array, truncating or padding with zeros (if necessary) so the copy has the specified length.
`static char[]`	**`copyOf`**`(char[] original, int newLength)`
	Copies the specified array, truncating or padding with null characters (if necessary) so the copy has the specified length.
`static double[]`	**`copyOf`**`(double[] original, int newLength)`
	Copies the specified array, truncating or padding with zeros (if necessary) so the copy has the specified length.

`static float[]`	**`copyOf`**`(float[] original, int newLength)` Copies the specified array, truncating or padding with zeros (if necessary) so the copy has the specified length.
`static int[]`	**`copyOf`**`(int[] original, int newLength)` Copies the specified array, truncating or padding with zeros (if necessary) so the copy has the specified length.
`static long[]`	**`copyOf`**`(long[] original, int newLength)` Copies the specified array, truncating or padding with zeros (if necessary) so the copy has the specified length.
`static short[]`	**`copyOf`**`(short[] original, int newLength)` Copies the specified array, truncating or padding with zeros (if necessary) so the copy has the specified length.
`static <T> T[]`	**`copyOf`**`(T[] original, int newLength)` Copies the specified array, truncating or padding with nulls (if necessary) so the copy has the specified length.
`static <T,U> T[]`	**`copyOf`**`(U[] original, int newLength, `**`Class`**`<? extends T[]> newType)` Copies the specified array, truncating or padding with nulls (if necessary) so the copy has the specified length.
`static boolean[]`	**`copyOfRange`**`(boolean[] original, int from, int to)` Copies the specified range of the specified array into a new array.
`static byte[]`	**`copyOfRange`**`(byte[] original, int from, int to)` Copies the specified range of the specified array into a new array.
`static char[]`	**`copyOfRange`**`(char[] original, int from, int to)` Copies the specified range of the specified array into a new array.
`static double[]`	**`copyOfRange`**`(double[] original, int from, int to)` Copies the specified range of the specified array into a new array.
`static float[]`	**`copyOfRange`**`(float[] original, int from, int to)` Copies the specified range of the specified array into a new array.
`static int[]`	**`copyOfRange`**`(int[] original, int from, int to)` Copies the specified range of the specified array into a new array.
`static long[]`	**`copyOfRange`**`(long[] original, int from, int to)` Copies the specified range of the specified array into a new array.
`static short[]`	**`copyOfRange`**`(short[] original, int from, int to)` Copies the specified range of the specified array into a new array.
`static <T> T[]`	**`copyOfRange`**`(T[] original, int from, int to)` Copies the specified range of the specified array into a new array.

`static <T,U> T[]`	`copyOfRange(U[] original, int from, int to, Class<? extends T[]> newType)` Copies the specified range of the specified array into a new array.
`static boolean`	`deepEquals(Object[] a1, Object[] a2)` Returns `true` if the two specified arrays are *deeply equal* to one another.
`static int`	`deepHashCode(Object[] a)` Returns a hash code based on the "deep contents" of the specified array.
`static String`	`deepToString(Object[] a)` Returns a string representation of the "deep contents" of the specified array.
`static boolean`	`equals(boolean[] a, boolean[] a2)` Returns `true` if the two specified arrays of booleans are *equal* to one another.
`static boolean`	`equals(byte[] a, byte[] a2)` Returns `true` if the two specified arrays of bytes are *equal* to one another.
`static boolean`	`equals(char[] a, char[] a2)` Returns `true` if the two specified arrays of chars are *equal* to one another.
`static boolean`	`equals(double[] a, double[] a2)` Returns `true` if the two specified arrays of doubles are *equal* to one another.
`static boolean`	`equals(float[] a, float[] a2)` Returns `true` if the two specified arrays of floats are *equal* to one another.
`static boolean`	`equals(int[] a, int[] a2)` Returns `true` if the two specified arrays of ints are *equal* to one another.
`static boolean`	`equals(long[] a, long[] a2)` Returns `true` if the two specified arrays of longs are *equal* to one another.
`static boolean`	`equals(Object[] a, Object[] a2)` Returns `true` if the two specified arrays of Objects are *equal* to one another.
`static boolean`	`equals(short[] a, short[] a2)`

	Returns `true` if the two specified arrays of shorts are *equal* to one another.
`static void`	**`fill`**`(boolean[] a, boolean val)` Assigns the specified boolean value to each element of the specified array of booleans.
`static void`	**`fill`**`(boolean[] a, int fromIndex, int toIndex, boolean val)` Assigns the specified boolean value to each element of the specified range of the specified array of booleans.
`static void`	**`fill`**`(byte[] a, byte val)` Assigns the specified byte value to each element of the specified array of bytes.
`static void`	**`fill`**`(byte[] a, int fromIndex, int toIndex, byte val)` Assigns the specified byte value to each element of the specified range of the specified array of bytes.
`static void`	**`fill`**`(char[] a, char val)` Assigns the specified char value to each element of the specified array of chars.
`static void`	**`fill`**`(char[] a, int fromIndex, int toIndex, char val)` Assigns the specified char value to each element of the specified range of the specified array of chars.
`static void`	**`fill`**`(double[] a, double val)` Assigns the specified double value to each element of the specified array of doubles.
`static void`	**`fill`**`(double[] a, int fromIndex, int toIndex, double val)` Assigns the specified double value to each element of the specified range of the specified array of doubles.
`static void`	**`fill`**`(float[] a, float val)` Assigns the specified float value to each element of the specified array of floats.
`static void`	**`fill`**`(float[] a, int fromIndex, int toIndex, float val)` Assigns the specified float value to each element of the specified range of the specified array of floats.
`static void`	**`fill`**`(int[] a, int val)` Assigns the specified int value to each element of the specified array of ints.

`static void`	**`fill`**`(int[] a, int fromIndex, int toIndex, int val)` Assigns the specified int value to each element of the specified range of the specified array of ints.
`static void`	**`fill`**`(long[] a, int fromIndex, int toIndex, long val)` Assigns the specified long value to each element of the specified range of the specified array of longs.
`static void`	**`fill`**`(long[] a, long val)` Assigns the specified long value to each element of the specified array of longs.
`static void`	**`fill`**`(`**`Object`**`[] a, int fromIndex, int toIndex, `**`Object`**` val)` Assigns the specified Object reference to each element of the specified range of the specified array of Objects.
`static void`	**`fill`**`(`**`Object`**`[] a, `**`Object`**` val)` Assigns the specified Object reference to each element of the specified array of Objects.
`static void`	**`fill`**`(short[] a, int fromIndex, int toIndex, short val)` Assigns the specified short value to each element of the specified range of the specified array of shorts.
`static void`	**`fill`**`(short[] a, short val)` Assigns the specified short value to each element of the specified array of shorts.
`static int`	**`hashCode`**`(boolean[] a)` Returns a hash code based on the contents of the specified array.
`static int`	**`hashCode`**`(byte[] a)` Returns a hash code based on the contents of the specified array.
`static int`	**`hashCode`**`(char[] a)` Returns a hash code based on the contents of the specified array.
`static int`	**`hashCode`**`(double[] a)` Returns a hash code based on the contents of the specified array.
`static int`	**`hashCode`**`(float[] a)` Returns a hash code based on the contents of the specified array.
`static int`	**`hashCode`**`(int[] a)` Returns a hash code based on the contents of the specified array.
`static int`	**`hashCode`**`(long[] a)` Returns a hash code based on the contents of the specified array.

`static int`	`hashCode(Object[] a)` Returns a hash code based on the contents of the specified array.
`static int`	`hashCode(short[] a)` Returns a hash code based on the contents of the specified array.
`static void`	`parallelPrefix(double[] array, DoubleBinaryOperator op)` Cumulates, in parallel, each element of the given array in place, using the supplied function.
`static void`	`parallelPrefix(double[] array, int fromIndex, int toIndex, DoubleBinaryOperator op)` Performs `parallelPrefix(double[], DoubleBinaryOperator)` for the given subrange of the array.
`static void`	`parallelPrefix(int[] array, IntBinaryOperator op)` Cumulates, in parallel, each element of the given array in place, using the supplied function.
`static void`	`parallelPrefix(int[] array, int fromIndex, int toIndex, IntBinaryOperator op)` Performs `parallelPrefix(int[], IntBinaryOperator)` for the given subrange of the array.
`static void`	`parallelPrefix(long[] array, int fromIndex, int toIndex, LongBinaryOperator op)` Performs `parallelPrefix(long[], LongBinaryOperator)` for the given subrange of the array.
`static void`	`parallelPrefix(long[] array, LongBinaryOperator op)` Cumulates, in parallel, each element of the given array in place, using the supplied function.
`static <T> void`	`parallelPrefix(T[] array, BinaryOperator<T> op)` Cumulates, in parallel, each element of the given array in place, using the supplied function.
`static <T> void`	`parallelPrefix(T[] array, int fromIndex, int toIndex, BinaryOperator<T> op)` Performs `parallelPrefix(Object[], BinaryOperator)` for the given subrange of the array.
`static void`	`parallelSetAll(double[] array, IntToDoubleFunction generator)` Set all elements of the specified array, in parallel, using the provided generator function to compute each element.
`static void`	`parallelSetAll(int[] array, IntUnaryOperator generator)`

	Set all elements of the specified array, in parallel, using the provided generator function to compute each element.
`static void`	**`parallelSetAll`**`(long[] array, `**`IntToLongFunction`**` generator)` Set all elements of the specified array, in parallel, using the provided generator function to compute each element.
`static <T> void`	**`parallelSetAll`**`(T[] array, `**`IntFunction`**`<? extends T> generator)` Set all elements of the specified array, in parallel, using the provided generator function to compute each element.
`static void`	**`parallelSort`**`(byte[] a)` Sorts the specified array into ascending numerical order.
`static void`	**`parallelSort`**`(byte[] a, int fromIndex, int toIndex)` Sorts the specified range of the array into ascending numerical order.
`static void`	**`parallelSort`**`(char[] a)` Sorts the specified array into ascending numerical order.
`static void`	**`parallelSort`**`(char[] a, int fromIndex, int toIndex)` Sorts the specified range of the array into ascending numerical order.
`static void`	**`parallelSort`**`(double[] a)` Sorts the specified array into ascending numerical order.
`static void`	**`parallelSort`**`(double[] a, int fromIndex, int toIndex)` Sorts the specified range of the array into ascending numerical order.
`static void`	**`parallelSort`**`(float[] a)` Sorts the specified array into ascending numerical order.
`static void`	**`parallelSort`**`(float[] a, int fromIndex, int toIndex)` Sorts the specified range of the array into ascending numerical order.
`static void`	**`parallelSort`**`(int[] a)` Sorts the specified array into ascending numerical order.
`static void`	**`parallelSort`**`(int[] a, int fromIndex, int toIndex)` Sorts the specified range of the array into ascending numerical order.
`static void`	**`parallelSort`**`(long[] a)` Sorts the specified array into ascending numerical order.
`static void`	**`parallelSort`**`(long[] a, int fromIndex, int toIndex)` Sorts the specified range of the array into ascending numerical order.
`static void`	**`parallelSort`**`(short[] a)` Sorts the specified array into ascending numerical order.

`static void`	**`parallelSort`**`(short[] a, int fromIndex, int toIndex)` Sorts the specified range of the array into ascending numerical order.
`static <T extends` **`Comparable`**`<? super T>> void`	**`parallelSort`**`(T[] a)` Sorts the specified array of objects into ascending order, according to the **natural ordering** of its elements.
`static <T> void`	**`parallelSort`**`(T[] a,` **`Comparator`**`<? super T> cmp)` Sorts the specified array of objects according to the order induced by the specified comparator.
`static <T extends` **`Comparable`**`<? super T>> void`	**`parallelSort`**`(T[] a, int fromIndex, int toIndex)` Sorts the specified range of the specified array of objects into ascending order, according to the **natural ordering** of its elements.
`static <T> void`	**`parallelSort`**`(T[] a, int fromIndex, int toIndex,` **`Comparator`**`<? super T> cmp)` Sorts the specified range of the specified array of objects according to the order induced by the specified comparator.
`static void`	**`setAll`**`(double[] array,` **`IntToDoubleFunction`** `generator)` Set all elements of the specified array, using the provided generator function to compute each element.
`static void`	**`setAll`**`(int[] array,` **`IntUnaryOperator`** `generator)` Set all elements of the specified array, using the provided generator function to compute each element.
`static void`	**`setAll`**`(long[] array,` **`IntToLongFunction`** `generator)` Set all elements of the specified array, using the provided generator function to compute each element.
`static <T> void`	**`setAll`**`(T[] array,` **`IntFunction`**`<? extends T> generator)` Set all elements of the specified array, using the provided generator function to compute each element.
`static void`	**`sort`**`(byte[] a)` Sorts the specified array into ascending numerical order.
`static void`	**`sort`**`(byte[] a, int fromIndex, int toIndex)` Sorts the specified range of the array into ascending order.

`static void`	**`sort`**`(char[] a)` Sorts the specified array into ascending numerical order.
`static void`	**`sort`**`(char[] a, int fromIndex, int toIndex)` Sorts the specified range of the array into ascending order.
`static void`	**`sort`**`(double[] a)` Sorts the specified array into ascending numerical order.
`static void`	**`sort`**`(double[] a, int fromIndex, int toIndex)` Sorts the specified range of the array into ascending order.
`static void`	**`sort`**`(float[] a)` Sorts the specified array into ascending numerical order.
`static void`	**`sort`**`(float[] a, int fromIndex, int toIndex)` Sorts the specified range of the array into ascending order.
`static void`	**`sort`**`(int[] a)` Sorts the specified array into ascending numerical order.
`static void`	**`sort`**`(int[] a, int fromIndex, int toIndex)` Sorts the specified range of the array into ascending order.
`static void`	**`sort`**`(long[] a)` Sorts the specified array into ascending numerical order.
`static void`	**`sort`**`(long[] a, int fromIndex, int toIndex)` Sorts the specified range of the array into ascending order.
`static void`	**`sort`**`(`**`Object`**`[] a)` Sorts the specified array of objects into ascending order, according to the **natural ordering** of its elements.
`static void`	**`sort`**`(`**`Object`**`[] a, int fromIndex, int toIndex)` Sorts the specified range of the specified array of objects into ascending order, according to the **natural ordering** of its elements.
`static void`	**`sort`**`(short[] a)` Sorts the specified array into ascending numerical order.
`static void`	**`sort`**`(short[] a, int fromIndex, int toIndex)` Sorts the specified range of the array into ascending order.
`static <T> void`	**`sort`**`(T[] a, `**`Comparator`**`<? super T> c)` Sorts the specified array of objects according to the order induced by the specified comparator.
`static <T> void`	**`sort`**`(T[] a, int fromIndex, int toIndex, `**`Comparator`**`<? super T> c)`

	Sorts the specified range of the specified array of objects according to the order induced by the specified comparator.
`static` **`Spliterator.OfDouble`**	**`spliterator`**`(double[] array)` Returns a **`Spliterator.OfDouble`** covering all of the specified array.
`static` **`Spliterator.OfDouble`**	**`spliterator`**`(double[] array, int startInclusive, int endExclusive)` Returns a **`Spliterator.OfDouble`** covering the specified range of the specified array.
`static` **`Spliterator.OfInt`**	**`spliterator`**`(int[] array)` Returns a **`Spliterator.OfInt`** covering all of the specified array.
`static` **`Spliterator.OfInt`**	**`spliterator`**`(int[] array, int startInclusive, int endExclusive)` Returns a **`Spliterator.OfInt`** covering the specified range of the specified array.
`static` **`Spliterator.OfLong`**	**`spliterator`**`(long[] array)` Returns a **`Spliterator.OfLong`** covering all of the specified array.
`static` **`Spliterator.OfLong`**	**`spliterator`**`(long[] array, int startInclusive, int endExclusive)` Returns a **`Spliterator.OfLong`** covering the specified range of the specified array.
`static <T>` **`Spliterator`**`<T>`	**`spliterator`**`(T[] array)` Returns a **`Spliterator`** covering all of the specified array.
`static <T>` **`Spliterator`**`<T>`	**`spliterator`**`(T[] array, int startInclusive, int endExclusive)` Returns a **`Spliterator`** covering the specified range of the specified array.
`static` **`DoubleStream`**	**`stream`**`(double[] array)` Returns a sequential **`DoubleStream`** with the specified array as its source.
`static` **`DoubleStream`**	**`stream`**`(double[] array, int startInclusive, int endExclusive)` Returns a sequential **`DoubleStream`** with the specified range of the specified array as its source.
`static` **`IntStream`**	**`stream`**`(int[] array)` Returns a sequential **`IntStream`** with the specified array as its source.

static **IntStream**	**stream**(int[] array, int startInclusive, int endExclusive) Returns a sequential **IntStream** with the specified range of the specified array as its source.
static **LongStream**	**stream**(long[] array) Returns a sequential **LongStream** with the specified array as its source.
static **LongStream**	**stream**(long[] array, int startInclusive, int endExclusive) Returns a sequential **LongStream** with the specified range of the specified array as its source.
static <T> **Stream**<T>	**stream**(T[] array) Returns a sequential **Stream** with the specified array as its source.
static <T> **Stream**<T>	**stream**(T[] array, int startInclusive, int endExclusive) Returns a sequential **Stream** with the specified range of the specified array as its source.
static **String**	**toString**(boolean[] a) Returns a string representation of the contents of the specified array.
static **String**	**toString**(byte[] a) Returns a string representation of the contents of the specified array.
static **String**	**toString**(char[] a) Returns a string representation of the contents of the specified array.
static **String**	**toString**(double[] a) Returns a string representation of the contents of the specified array.
static **String**	**toString**(float[] a) Returns a string representation of the contents of the specified array.
static **String**	**toString**(int[] a) Returns a string representation of the contents of the specified array.
static **String**	**toString**(long[] a) Returns a string representation of the contents of the specified array.
static **String**	**toString**(**Object**[] a) Returns a string representation of the contents of the specified array.
static **String**	**toString**(short[] a) Returns a string representation of the contents of the specified array.

Methods inherited from class java.lang.Object

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