public final class
String
extends Object
implements
Serializable CharSequence Comparable<T>

Summary: Fields | Ctors | Methods | Inherited Methods | [Expand All]
Added in API level 1

java.lang.Object
↳java.lang.String

Class Overview

An immutable sequence of characters/code units (`chars`). A `String` is represented by array of UTF-16 values, such that Unicode supplementary characters (code points) are stored/encoded as surrogate pairs via Unicode code units (`char`). ()

Backing Arrays

This class is implemented using a char[]. The length of the array may exceed the length of the string. For example, the string "Hello" may be backed by the array `['H', 'e', 'l', 'l', 'o', 'W'. 'o', 'r', 'l', 'd']` with offset 0 and length 5.

Multiple strings can share the same char[] because strings are immutable. The `substring(int)` (/reference/java/lang/String.html#substring(int)) method **always** returns a string that shares the backing array of its source string. Generally this is an optimization: fewer character arrays need to be allocated, and less copying is necessary. But this can also lead to unwanted heap retention. Taking a short substring of long string means that the long shared char[] won't be garbage until both strings are garbage. This typically happens when parsing small substrings out of a large input. To avoid this where necessary, call `new String(longString.subString(...))`. The string copy constructor always ensures that the backing array is no larger than necessary.

See Also
`StringBuffer`
`StringBuilder`
`Charset`

Summary

Fields	
public static final Comparator<String> CASE_INSENSITIVE_ORDER	A comparator ignoring the case of the

characters.

Public Constructors

String ()

Creates an empty string.

String (byte[] data)

Converts the byte array to a string using the system's `default charset`.

String (byte[] data, int high)

This constructor was deprecated in API level 1. Use `String(byte[])` or `String(byte[], String)` instead.

String (byte[] data, int offset, int byteCount)

Converts a subsequence of the byte array to a string using the system's `default charset`.

String (byte[] data, int high, int offset, int byteCount)

This constructor was deprecated in API level 1. Use `String(byte[], int, int)` instead.

String (byte[] data, int offset, int byteCount, String charsetName)

Converts the byte array to a string using the named charset.

String (byte[] data, String charsetName)

Converts the byte array to a string using the named charset.

String (byte[] data, int offset, int byteCount, Charset charset)

Converts the byte array to a string using the given charset.

String (byte[] data, Charset charset)

Converts the byte array to a String using the given charset.

String (char[] data)

Initializes this string to contain the characters in the specified character array.

String (char[] data, int offset, int charCount)

Initializes this string to contain the specified characters in the character array.

String (String toCopy)

Constructs a new string with the same sequence of characters as `toCopy`.

String (StringBuffer stringBuffer)

Creates a `String` from the contents of the specified `StringBuffer`.

String (int[] codePoints, int offset, int count)

Creates a `String` from the sub-array of Unicode code points.

String (StringBuilder stringBuilder)

Creates a `String` from the contents of the specified `StringBuilder`.

Public Methods

char	charAt (int index) Returns the character at the specified offset in this string.

int	codePointAt (int index) Returns the Unicode code point at the given `index`.
int	codePointBefore (int index) Returns the Unicode code point that precedes the given `index`.
int	codePointCount (int start, int end) Calculates the number of Unicode code points between `start` and `end`.
int	compareTo (String string) Compares the specified string to this string using the Unicode values of the characters.
int	compareToIgnoreCase (String string) Compares the specified string to this string using the Unicode values of the characters, ignoring case differences.
String	concat (String string) Concatenates this string and the specified string.
boolean	contains (CharSequence cs) Determines if this `String` contains the sequence of characters in the `CharSequence` passed.
boolean	contentEquals (CharSequence cs) Compares a `CharSequence` to this `String` to determine if their contents are equal.
boolean	contentEquals (StringBuffer strbuf) Returns whether the characters in the StringBuffer `strbuf` are the same as those in this string.
static String	copyValueOf (char[] data, int start, int length) Creates a new string containing the specified characters in the character array.
static String	copyValueOf (char[] data) Creates a new string containing the characters in the specified character array.
boolean	endsWith (String suffix) Compares the specified string to this string to determine if the specified string is a suffix.
boolean	equals (Object object) Compares the specified object to this string and returns true if they are equal.
boolean	equalsIgnoreCase (String string) Compares the specified string to this string ignoring the case of the characters and returns true if they are equal.
static String	format (Locale locale, String format, Object... args) Returns a formatted string, using the supplied format and arguments, localized to the given locale.
static String	format (String format, Object... args) Returns a localized formatted string, using the supplied format and arguments, using the user's default locale.

void	getBytes (int start, int end, byte[] data, int index) *This method was deprecated in API level 1. Use `getBytes()` or `getBytes(String)` instead.*
byte[]	getBytes (String charsetName) Returns a new byte array containing the characters of this string encoded using the named charset.
byte[]	getBytes (Charset charset) Returns a new byte array containing the characters of this string encoded using the given charset.
byte[]	getBytes () Returns a new byte array containing the characters of this string encoded using the system's `default charset`.
void	getChars (int start, int end, char[] buffer, int index) Copies the specified characters in this string to the character array starting at the specified offset in the character array.
int	hashCode () Returns an integer hash code for this object.
int	indexOf (int c) Searches in this string for the first index of the specified character.
int	indexOf (int c, int start) Searches in this string for the index of the specified character.
int	indexOf (String subString, int start) Searches in this string for the index of the specified string.
int	indexOf (String string) Searches in this string for the first index of the specified string.
String	intern () Returns an interned string equal to this string.
boolean	isEmpty () Returns true if the length of this string is 0.
int	lastIndexOf (String string) Searches in this string for the last index of the specified string.
int	lastIndexOf (int c, int start) Returns the last index of the code point `c`, or -1.
int	lastIndexOf (int c) Returns the last index of the code point `c`, or -1.
int	lastIndexOf (String subString, int start) Searches in this string for the index of the specified string.
int	length () Returns the number of characters in this string.
boolean	matches (String regularExpression) Tests whether this string matches the given `regularExpression`.
int	offsetByCodePoints (int index, int codePointOffset) Returns the index within this object that is offset from `index` by `codePointOffset` code points.

boolean	regionMatches (boolean ignoreCase, int thisStart, String string, int start, int length) Compares the specified string to this string and compares the specified range of characters to determine if they are the same.
boolean	regionMatches (int thisStart, String string, int start, int length) Compares the specified string to this string and compares the specified range of characters to determine if they are the same.
String	replace (CharSequence target, CharSequence replacement) Copies this string replacing occurrences of the specified target sequence with another sequence.
String	replace (char oldChar, char newChar) Copies this string replacing occurrences of the specified character with another character.
String	replaceAll (String regularExpression, String replacement) Replaces all matches for `regularExpression` within this string with the given `replacement`.
String	replaceFirst (String regularExpression, String replacement) Replaces the first match for `regularExpression` within this string with the given `replacement`.
String[]	split (String regularExpression) Splits this string using the supplied `regularExpression`.
String[]	split (String regularExpression, int limit) Splits this string using the supplied `regularExpression`.
boolean	startsWith (String prefix) Compares the specified string to this string to determine if the specified string is a prefix.
boolean	startsWith (String prefix, int start) Compares the specified string to this string, starting at the specified offset, to determine if the specified string is a prefix.
CharSequence	subSequence (int start, int end) Has the same result as the substring function, but is present so that string may implement the CharSequence interface.
String	substring (int start) Returns a string containing a suffix of this string.
String	substring (int start, int end) Returns a string containing a subsequence of characters from this string.
char[]	toCharArray () Returns a new `char` array containing a copy of the characters in this string.
String	toLowerCase (Locale locale) Converts this string to lower case, using the rules of `locale`.
String	toLowerCase () Converts this string to lower case, using the rules of the user's default locale.
String	toString () Returns this string.

String	toUpperCase (Locale locale) Converts this this string to upper case, using the rules of `locale`.
String	toUpperCase () Converts this this string to upper case, using the rules of the user's default locale.
String	trim () Copies this string removing white space characters from the beginning and end of the string.
static String	valueOf (long value) Converts the specified long to its string representation.
static String	valueOf (Object value) Converts the specified object to its string representation.
static String	valueOf (char[] data) Creates a new string containing the characters in the specified character array.
static String	valueOf (double value) Converts the specified double to its string representation.
static String	valueOf (int value) Converts the specified integer to its string representation.
static String	valueOf (float value) Converts the specified float to its string representation.
static String	valueOf (char[] data, int start, int length) Creates a new string containing the specified characters in the character array.
static String	valueOf (boolean value) Converts the specified boolean to its string representation.
static String	valueOf (char value) Converts the specified character to its string representation.

Inherited Methods [Expand]

- From class java.lang.Object
- From interface java.lang.CharSequence
- From interface java.lang.Comparable

Fields

public static final Comparator<String>
CASE_INSENSITIVE_ORDER Added in API level 1

A comparator ignoring the case of the characters.

Public Constructors

public **String** () Added in API level 1

Creates an empty string.

public **String** (byte[] data) Added in API level 1

Converts the byte array to a string using the system's `default charset` (/reference/java/nio/charset/Charset.html#defaultCharset()).

public **String** (byte[] data, int high) Added in API level 1

> **This constructor was deprecated in API level 1.**
> Use `String(byte[])` (/reference/java/lang/String.html#String(byte[])) or `String(byte[], String)` (/reference/java/lang/String.html#String(byte[], java.lang.String)) instead.

Converts the byte array to a string, setting the high byte of every character to the specified value.

Parameters

data	the byte array to convert to a string.
high	the high byte to use.

Throws

NullPointerException	if `data == null`.

public **String** (byte[] data, int offset, int byteCount) Added in API level 1

Converts a subsequence of the byte array to a string using the system's `default charset` (/reference/java/nio/charset/Charset.html#defaultCharset()).

Throws

NullPointerException	if `data == null`.
IndexOutOfBoundsException	if `byteCount < 0 || offset < 0 || offset + byteCount > data.length`.

public **String** (byte[] data, int high, int offset, int byteCount) Added in API level 1

> **This constructor was deprecated in API level 1.**
> Use `String(byte[], int, int)` (/reference/java/lang/String.html#String(byte[], int, int)) instead.

Converts the byte array to a string, setting the high byte of every character to `high`.

Throws

NullPointerException	if `data == null`.
IndexOutOfBoundsException	if `byteCount < 0 || offset < 0 || offset + byteCount > data.length`

public **String** (byte[] data, int offset, int byteCount, String charsetName)

Added in API level 1

Converts the byte array to a string using the named charset.

The behavior when the bytes cannot be decoded by the named charset is unspecified. Use `CharsetDecoder (/reference/java/nio/charset/CharsetDecoder.html)` for more control.

Throws

NullPointerException	if `data == null`.
IndexOutOfBoundsException	if `byteCount < 0 || offset < 0 || offset + byteCount > data.length`.
UnsupportedEncodingException	if the named charset is not supported.

public **String** (byte[] data, String charsetName)

Added in API level 1

Converts the byte array to a string using the named charset.

The behavior when the bytes cannot be decoded by the named charset is unspecified. Use `CharsetDecoder (/reference/java/nio/charset/CharsetDecoder.html)` for more control.

Throws

NullPointerException	if `data == null`.
UnsupportedEncodingException	if `charsetName` is not supported.

public **String** (byte[] data, int offset, int byteCount, Charset charset)

Added in API level 9

Converts the byte array to a string using the given charset.

The behavior when the bytes cannot be decoded by the given charset is to replace malformed input and unmappable characters with the charset's default replacement string. Use `CharsetDecoder (/reference/java/nio/charset/CharsetDecoder.html)` for more control.

Throws

IndexOutOfBoundsException	if `byteCount < 0 || offset < 0 || offset + byteCount > data.length`

NullPointerException	if `data == null`

public **String** (byte[] data, Charset charset) Added in API level 9

Converts the byte array to a String using the given charset.

Throws

NullPointerException	if `data == null`

public **String** (char[] data) Added in API level 1

Initializes this string to contain the characters in the specified character array. Modifying the character array after creating the string has no effect on the string.

Throws

NullPointerException	if `data == null`

public **String** (char[] data, int offset, int charCount) Added in API level 1

Initializes this string to contain the specified characters in the character array. Modifying the character array after creating the string has no effect on the string.

Throws

NullPointerException	if `data == null`.
IndexOutOfBoundsException	if `charCount < 0 || offset < 0 || offset + charCount > data.length`

public **String** (String toCopy) Added in API level 1

Constructs a new string with the same sequence of characters as `toCopy`. The returned string's backing array (#backing_array) is no larger than necessary.

public **String** (StringBuffer stringBuffer) Added in API level 1

Creates a `String` from the contents of the specified `StringBuffer`.

public **String** (int[] codePoints, int offset, int count) Added in API level 1

Creates a `String` from the sub-array of Unicode code points.

Throws

NullPointerException	if `codePoints == null`.
IllegalArgumentException	if any of the elements of `codePoints` are not valid Unicode code points.

IndexOutOfBoundsException	if `offset` or `count` are not within the bounds of `codePoints`.

public **String** (StringBuilder stringBuilder)

Added in API level 1

Creates a `String` from the contents of the specified `StringBuilder`.

Throws

NullPointerException	if `stringBuilder == null`.

Public Methods

public char **charAt** (int index)

Added in API level 1

Returns the character at the specified offset in this string.

Parameters

index	the zero-based index in this string.

Returns

the character at the index.

Throws

IndexOutOfBoundsException	if `index < 0` or `index >= length()`.

public int **codePointAt** (int index)

Added in API level 1

Returns the Unicode code point at the given `index`.

Throws

IndexOutOfBoundsException	if `index < 0 || index >= length()`

See Also

`codePointAt(char[], int, int)`

public int **codePointBefore** (int index)

Added in API level 1

Returns the Unicode code point that precedes the given `index`.

Throws

IndexOutOfBoundsException	if `index < 1 || index > length()`

See Also

`codePointBefore(char[], int, int)`

public int **codePointCount** (int start, int end)

Added in API level 1

Calculates the number of Unicode code points between `start` and `end`.

Parameters

start	the inclusive beginning index of the subsequence.
end	the exclusive end index of the subsequence.

Returns

the number of Unicode code points in the subsequence.

Throws

IndexOutOfBoundsException	if `start < 0 || end > length() || start > end`

See Also

`codePointCount(CharSequence, int, int)`

public int **compareTo** (String string)

Added in API level 1

Compares the specified string to this string using the Unicode values of the characters. Returns 0 if the strings contain the same characters in the same order. Returns a negative integer if the first non-equal character in this string has a Unicode value which is less than the Unicode value of the character at the same position in the specified string, or if this string is a prefix of the specified string. Returns a positive integer if the first non-equal character in this string has a Unicode value which is greater than the Unicode value of the character at the same position in the specified string, or if the specified string is a prefix of this string.

Parameters

string	the string to compare.

Returns

0 if the strings are equal, a negative integer if this string is before the specified string, or a positive integer if this string is after the specified string.

Throws

NullPointerException	if `string` is `null`.

public int **compareToIgnoreCase** (String string)

Added in API level 1

Compares the specified string to this string using the Unicode values of the characters, ignoring case differences. Returns 0 if the strings contain the same characters in the same order. Returns a negative integer if the first non-equal character in this string has a Unicode value which is less than the Unicode value of the character at the same position in the specified string, or if this string is a prefix of the specified string. Returns a positive integer if the first non-equal character in this string has a Unicode value which is greater than the Unicode value of the character at the same position in the specified string, or if the specified string is a prefix of this string.

Parameters

string	the string to compare.

Returns

0 if the strings are equal, a negative integer if this string is before the specified string, or a positive integer if this string is after the specified string.

Throws

NullPointerException	if `string` is `null`.

public String **concat** (String string)

Added in API level 1

Concatenates this string and the specified string.

Parameters

string	the string to concatenate

Returns

a new string which is the concatenation of this string and the specified string.

public boolean **contains** (CharSequence cs)

Added in API level 1

Determines if this `String` contains the sequence of characters in the `CharSequence` passed.

Parameters

cs	the character sequence to search for.

Returns

`true` if the sequence of characters are contained in this string, otherwise `false`.

public boolean **contentEquals** (CharSequence cs)

Added in API level 1

Compares a `CharSequence` to this `String` to determine if their contents are equal.

Parameters

cs	the character sequence to compare to.

Returns

`true` if equal, otherwise `false`

public boolean **contentEquals** (StringBuffer strbuf)

Added in API level 1

Returns whether the characters in the StringBuffer `strbuf` are the same as those in this string.

Parameters

strbuf	the StringBuffer to compare this string to.

Returns

`true` if the characters in `strbuf` are identical to those in this string. If they are not, `false` will be returned.

Throws

NullPointerException	if `strbuf` is `null`.

public static String **copyValueOf** (char[] data, int start, int length)

Added in API level 1

Creates a new string containing the specified characters in the character array. Modifying the character array after creating the string has no effect on the string.

Parameters

data	the array of characters.
start	the starting offset in the character array.
length	the number of characters to use.

Returns

the new string.

Throws

NullPointerException	if `data` is `null`.
IndexOutOfBoundsException	if `length < 0, start < 0` or `start + length > data.length`.

public static String **copyValueOf** (char[] data)

Added in API level 1

Creates a new string containing the characters in the specified character array. Modifying the character array after creating the string has no effect on the string.

Parameters

data	the array of characters.

Returns

the new string.

Throws

NullPointerException	if `data` is `null`.

public boolean **endsWith** (String suffix)

Added in API level 1

Compares the specified string to this string to determine if the specified string is a suffix.

Parameters

suffix	the suffix to look for.

Returns

`true` if the specified string is a suffix of this string, `false` otherwise.

Throws

NullPointerException	if `suffix` is `null`.

public boolean **equals** (Object object)

Added in API level 1

Compares the specified object to this string and returns true if they are equal. The object must be an instance of string with the same characters in the same order.

Parameters

object	the object to compare.

Returns

`true` if the specified object is equal to this string, `false` otherwise.

See Also

`hashCode()`

public boolean **equalsIgnoreCase** (String string)

Added in API level 1

Compares the specified string to this string ignoring the case of the characters and returns true if they are equal.

Parameters

string	the string to compare.

Returns

`true` if the specified string is equal to this string, `false` otherwise.

public static String **format** (Locale locale, String format, Object... args)

Added in API level 1

Returns a formatted string, using the supplied format and arguments, localized to the given locale.

Parameters

locale	the locale to apply; `null` value means no localization.
format	the format string (see `format(String, Object...)`)
args	the list of arguments passed to the formatter. If there are more arguments than required by `format`, additional arguments are ignored.

Returns

the formatted string.

Throws

NullPointerException	if `format == null`
IllegalFormatException	if the format is invalid.

public static String **format** (String format, Object... args) Added in API level 1

Returns a localized formatted string, using the supplied format and arguments, using the user's default locale.

If you're formatting a string other than for human consumption, you should use the `format(Locale, String, Object...)` overload and supply `Locale.US`. See "Be wary of the default locale (../util/Locale.html#default_locale)".

Parameters

format	the format string (see `format(String, Object...)`)
args	the list of arguments passed to the formatter. If there are more arguments than required by `format`, additional arguments are ignored.

Returns
the formatted string.

Throws

NullPointerException	if `format == null`
IllegalFormatException	if the format is invalid.

public void **getBytes** (int start, int end, byte[] data, int index) Added in API level 1

This method was deprecated in API level 1.
Use `getBytes()` `(/reference/java/lang/String.html#getBytes())` or `getBytes(String)` `(/reference/java/lang/String.html#getBytes(java.lang.String))` instead.

Mangles this string into a byte array by stripping the high order bits from each character. Use `getBytes()` `(/reference/java/lang/String.html#getBytes())` or `getBytes(String)` `(/reference/java/lang/String.html#getBytes(java.lang.String))` instead.

Parameters

start	the starting offset of characters to copy.
end	the ending offset of characters to copy.
data	the destination byte array.
index	the starting offset in the destination byte array.

Throws

NullPointerException	if `data` is `null`.
IndexOutOfBoundsException	if `start < 0`, `end > length()`, `index < 0` or `end - start > data.length - index`.

public byte[] **getBytes** (String charsetName)

Added in API level 1

Returns a new byte array containing the characters of this string encoded using the named charset.

The behavior when this string cannot be represented in the named charset is unspecified. Use `CharsetEncoder` (/reference/java/nio/charset/CharsetEncoder.html) for more control.

Throws

UnsupportedEncodingException	if the charset is not supported

public byte[] **getBytes** (Charset charset)

Added in API level 9

Returns a new byte array containing the characters of this string encoded using the given charset.

The behavior when this string cannot be represented in the given charset is to replace malformed input and unmappable characters with the charset's default replacement byte array. Use `CharsetEncoder` (/reference/java/nio/charset/CharsetEncoder.html) for more control.

public byte[] **getBytes** ()

Added in API level 1

Returns a new byte array containing the characters of this string encoded using the system's `default charset` (/reference/java/nio/charset/Charset.html#defaultCharset()).

The behavior when this string cannot be represented in the system's default charset is unspecified. In practice, when the default charset is UTF-8 (as it is on Android), all strings can be encoded.

public void **getChars** (int start, int end, char[] buffer, int index)

Added in API level 1

Copies the specified characters in this string to the character array starting at the specified offset in the character array.

Parameters

start	the starting offset of characters to copy.
end	the ending offset of characters to copy.
buffer	the destination character array.
index	the starting offset in the character array.

Throws

NullPointerException	if `buffer` is `null`.
IndexOutOfBoundsException	if `start < 0`, `end > length()`, `start > end`, `index < 0`, `end - start > buffer.length - index`

public int **hashCode** ()

Added in API level 1

Returns an integer hash code for this object. By contract, any two objects for which `equals(Object)` (/reference/java/lang/Object.html#equals(java.lang.Object)) returns `true` must return the same hash code value. This means that subclasses of `Object` usually override both methods or neither method.

Note that hash values must not change over time unless information used in equals comparisons also changes.

See Writing a correct `hashCode` method (/reference/java/lang/Object.html#writing_hashCode) if you intend implementing your own `hashCode` method.

Returns

this object's hash code.

public int **indexOf** (int c)

Added in API level 1

Searches in this string for the first index of the specified character. The search for the character starts at the beginning and moves towards the end of this string.

Parameters

c	the character to find.

Returns

the index in this string of the specified character, -1 if the character isn't found.

public int **indexOf** (int c, int start)

Added in API level 1

Searches in this string for the index of the specified character. The search for the character starts at the specified offset and moves towards the end of this string.

Parameters

c	the character to find.
start	the starting offset.

Returns

the index in this string of the specified character, -1 if the character isn't found.

public int **indexOf** (String subString, int start)

Added in API level 1

Searches in this string for the index of the specified string. The search for the string starts at the specified offset and moves towards the end of this string.

Parameters

subString	the string to find.
start	the starting offset.

Returns

the index of the first character of the specified string in this string, -1 if the specified string is not a substring.

Throws

NullPointerException	if `subString` is `null`.

public int **indexOf** (String string)

Added in API level 1

Searches in this string for the first index of the specified string. The search for the string starts at the beginning and moves towards the end of this string.

Parameters

string	the string to find.

Returns

the index of the first character of the specified string in this string, -1 if the specified string is not a substring.

Throws

NullPointerException	if `string` is `null`.

public String **intern** ()

Added in API level 1

Returns an interned string equal to this string. The VM maintains an internal set of unique strings. All string literals found in loaded classes' constant pools are automatically interned. Manually-interned strings are only weakly referenced, so calling `intern` won't lead to unwanted retention.

Interning is typically used because it guarantees that for interned strings a and b, `a.equals(b)` can be simplified to `a == b`. (This is not true of non-interned strings.)

Many applications find it simpler and more convenient to use an explicit `HashMap (/reference/java/util/HashMap.html)` to implement their own pools.

public boolean **isEmpty** ()

Added in API level 9

Returns true if the length of this string is 0.

public int **lastIndexOf** (String string)

Added in API level 1

Searches in this string for the last index of the specified string. The search for the string starts at the end and moves towards the beginning of this string.

Parameters

string	the string to find.

Returns

the index of the first character of the specified string in this string, -1 if the specified string is not a substring.

Throws

NullPointerException	if `string` is `null`.

public int **lastIndexOf** (int c, int start)

Added in API level 1

Returns the last index of the code point `c`, or -1. The search for the character starts at offset `start` and moves towards the beginning of this string.

public int **lastIndexOf** (int c)

Added in API level 1

Returns the last index of the code point `c`, or -1. The search for the character starts at the end and moves towards the beginning of this string.

public int **lastIndexOf** (String subString, int start)

Added in API level 1

Searches in this string for the index of the specified string. The search for the string starts at the specified offset and moves towards the beginning of this string.

Parameters

subString	the string to find.
start	the starting offset.

Returns

the index of the first character of the specified string in this string , -1 if the specified string is not a substring.

Throws

NullPointerException	if `subString` is `null`.

public int **length** ()

Added in API level 1

Returns the number of characters in this string.

Returns

the number of characters.

public boolean **matches** (String regularExpression) Added in API level 1

Tests whether this string matches the given `regularExpression`. This method returns true only if the regular expression matches the *entire* input string. A common mistake is to assume that this method behaves like `contains(CharSequence)` (/reference/java/lang/String.html#contains(java.lang.CharSequence)); if you want to match anywhere within the input string, you need to add `.*` to the beginning and end of your regular expression. See `matches(String, CharSequence)` (/reference/java/util/regex/Pattern.html#matches(java.lang.String, java.lang.CharSequence)).

If the same regular expression is to be used for multiple operations, it may be more efficient to reuse a compiled `Pattern`.

Throws

	if the syntax of the supplied regular expression is not valid.
NullPointerException	if `regularExpression == null`

public int **offsetByCodePoints** (int index, int codePointOffset) Added in API level 1

Returns the index within this object that is offset from `index` by `codePointOffset` code points.

Parameters

index	the index within this object to calculate the offset from.
codePointOffset	the number of code points to count.

Returns

the index within this object that is the offset.

Throws

IndexOutOfBoundsException	if `index` is negative or greater than `length()` or if there aren't enough code points before or after `index` to match `codePointOffset`.

public boolean **regionMatches** (boolean ignoreCase, int thisStart, String string, int start, int length) Added in API level 1

Compares the specified string to this string and compares the specified range of characters to determine if they are the same. When ignoreCase is true, the case of the characters is ignored during the comparison.

Parameters

ignoreCase	specifies if case should be ignored.
thisStart	the starting offset in this string.
string	the string to compare.
start	the starting offset in the specified string.
length	the number of characters to compare.

Returns

`true` if the ranges of characters are equal, `false` otherwise.

Throws

NullPointerException	if `string` is `null`.

public boolean **regionMatches** (int thisStart, String string, int start, int length)

Added in API level 1

Compares the specified string to this string and compares the specified range of characters to determine if they are the same.

Parameters

thisStart	the starting offset in this string.
string	the string to compare.
start	the starting offset in the specified string.
length	the number of characters to compare.

Returns

`true` if the ranges of characters are equal, `false` otherwise

Throws

NullPointerException	if `string` is `null`.

public String **replace** (CharSequence target, CharSequence replacement)

Added in API level 1

Copies this string replacing occurrences of the specified target sequence with another sequence. The string is processed from the beginning to the end.

Parameters

target	the sequence to replace.
replacement	the replacement sequence.

Returns

the resulting string.

Throws

NullPointerException	if `target` or `replacement` is `null`.

public String **replace** (char oldChar, char newChar) Added in API level 1

Copies this string replacing occurrences of the specified character with another character.

Parameters

oldChar	the character to replace.
newChar	the replacement character.

Returns

a new string with occurrences of oldChar replaced by newChar.

public String **replaceAll** (String regularExpression, String replacement) Added in API level 1

Replaces all matches for `regularExpression` within this string with the given `replacement`. See `Pattern (/reference/java/util/regex/Pattern.html)` for regular expression syntax.

If the same regular expression is to be used for multiple operations, it may be more efficient to reuse a compiled `Pattern`.

Throws

	if the syntax of the supplied regular expression is not valid.
NullPointerException	if `regularExpression == null`

See Also

`Pattern`

public String **replaceFirst** (String regularExpression, String replacement) Added in API level 1

Replaces the first match for `regularExpression` within this string with the given `replacement`. See `Pattern (/reference/java/util/regex/Pattern.html)` for regular expression syntax.

If the same regular expression is to be used for multiple operations, it may be more efficient to reuse a compiled `Pattern`.

Throws

	if the syntax of the supplied regular expression is not valid.
NullPointerException	if `regularExpression == null`

See Also

`Pattern`

public String[] **split** (String regularExpression) Added in API level 1

Splits this string using the supplied `regularExpression`. Equivalent to `split(regularExpression, 0)`. See `split(CharSequence, int)` (/reference/java/util/regex/Pattern.html#split(java.lang.CharSequence, int)) for an explanation of `limit`. See `Pattern` (/reference/java/util/regex/Pattern.html) for regular expression syntax.

If the same regular expression is to be used for multiple operations, it may be more efficient to reuse a compiled `Pattern`.

Throws

NullPointerException	if `regularExpression == null`
	if the syntax of the supplied regular expression is not valid.

See Also

`Pattern`

public String[] **split** (String regularExpression, int limit) Added in API level 1

Splits this string using the supplied `regularExpression`. See `split(CharSequence, int)` (/reference/java/util/regex/Pattern.html#split(java.lang.CharSequence, int)) for an explanation of `limit`. See `Pattern` (/reference/java/util/regex/Pattern.html) for regular expression syntax.

If the same regular expression is to be used for multiple operations, it may be more efficient to reuse a compiled `Pattern`.

Throws

NullPointerException	if `regularExpression == null`
	if the syntax of the supplied regular expression is not valid.

public boolean **startsWith** (String prefix) Added in API level 1

Compares the specified string to this string to determine if the specified string is a prefix.

Parameters

prefix	the string to look for.

Returns

`true` if the specified string is a prefix of this string, `false` otherwise

Throws

NullPointerException	if `prefix` is `null`.

public boolean **startsWith** (String prefix, int start)

Added in API level 1

Compares the specified string to this string, starting at the specified offset, to determine if the specified string is a prefix.

Parameters

prefix	the string to look for.
start	the starting offset.

Returns

`true` if the specified string occurs in this string at the specified offset, `false` otherwise.

Throws

NullPointerException	if `prefix` is `null`.

public CharSequence **subSequence** (int start, int end)

Added in API level 1

Has the same result as the substring function, but is present so that string may implement the CharSequence interface.

Parameters

start	the offset the first character.
end	the offset of one past the last character to include.

Returns

the subsequence requested.

Throws

IndexOutOfBoundsException	if `start < 0`, `end < 0`, `start > end` or `end > length()`.

See Also

`subSequence(int, int)`

public String **substring** (int start)

Added in API level 1

Returns a string containing a suffix of this string. The returned string shares this string's backing array (#backing_array).

Parameters

start	the offset of the first character.

Returns

a new string containing the characters from start to the end of the string.

Throws

IndexOutOfBoundsException	if `start < 0` or `start > length()`.

public String **substring** (int start, int end)
Added in API level 1

Returns a string containing a subsequence of characters from this string. The returned string shares this string's backing array (#backing_array).

Parameters

start	the offset of the first character.
end	the offset one past the last character.

Returns

a new string containing the characters from start to end - 1

Throws

IndexOutOfBoundsException	if `start < 0`, `start > end` or `end > length()`.

public char[] **toCharArray** ()
Added in API level 1

Returns a new `char` array containing a copy of the characters in this string. This is expensive and rarely useful. If you just want to iterate over the characters in the string, use `charAt(int)` `(/reference/java/lang/String.html#charAt(int))` instead.

public String **toLowerCase** (Locale locale)
Added in API level 1

Converts this string to lower case, using the rules of `locale`.

Most case mappings are unaffected by the language of a `Locale`. Exceptions include dotted and dotless I in Azeri and Turkish locales, and dotted and dotless I and J in Lithuanian locales. On the other hand, it isn't necessary to provide a Greek locale to get correct case mapping of Greek characters: any locale will do.

See http://www.unicode.org/Public/UNIDATA/SpecialCasing.txt (http://www.unicode.org/Public/UNIDATA/SpecialCasing.txt) for full details of context- and language-specific special cases.

Returns

a new lower case string, or `this` if it's already all lower case.

public String **toLowerCase** ()
Added in API level 1

Converts this string to lower case, using the rules of the user's default locale. See "Be wary of the default locale (../util/Locale.html#default_locale)".

Returns

a new lower case string, or `this` if it's already all lower case.

public String **toString** ()
Added in API level 1

Returns this string.

Returns

a printable representation of this object.

public String **toUpperCase** (Locale locale)

Added in API level 1

Converts this this string to upper case, using the rules of `locale`.

Most case mappings are unaffected by the language of a `Locale`. Exceptions include dotted and dotless I in Azeri and Turkish locales, and dotted and dotless I and J in Lithuanian locales. On the other hand, it isn't necessary to provide a Greek locale to get correct case mapping of Greek characters: any locale will do.

See http://www.unicode.org/Public/UNIDATA/SpecialCasing.txt (http://www.unicode.org/Public/UNIDATA/SpecialCasing.txt) for full details of context- and language-specific special cases.

Returns

a new upper case string, or `this` if it's already all upper case.

public String **toUpperCase** ()

Added in API level 1

Converts this this string to upper case, using the rules of the user's default locale. See "Be wary of the default locale (../util/Locale.html#default_locale)".

Returns

a new upper case string, or `this` if it's already all upper case.

public String **trim** ()

Added in API level 1

Copies this string removing white space characters from the beginning and end of the string.

Returns

a new string with characters `<= \\u0020` removed from the beginning and the end.

public static String **valueOf** (long value)

Added in API level 1

Converts the specified long to its string representation.

Parameters

value	the long.

Returns

the long converted to a string.

public static String **valueOf** (Object value)

Added in API level 1

Converts the specified object to its string representation. If the object is null return the string `"null"`, otherwise use `toString()` to get the string representation.

Parameters

value	the object.

Returns

the object converted to a string, or the string `"null"`.

public static String **valueOf** (char[] data)

Added in API level 1

Creates a new string containing the characters in the specified character array. Modifying the character array after creating the string has no effect on the string.

Parameters

data	the array of characters.

Returns

the new string.

Throws

NullPointerException	if `data` is `null`.

public static String **valueOf** (double value)

Added in API level 1

Converts the specified double to its string representation.

Parameters

value	the double.

Returns

the double converted to a string.

public static String **valueOf** (int value)

Added in API level 1

Converts the specified integer to its string representation.

Parameters

value	the integer.

Returns

the integer converted to a string.

public static String **valueOf** (float value)

Added in API level 1

Converts the specified float to its string representation.

Parameters

value	the float.

Returns

the float converted to a string.

public static String **valueOf** (char[] data, int start, int length)

Added in API level 1

Creates a new string containing the specified characters in the character array. Modifying the character array after creating the string has no effect on the string.

Parameters

data	the array of characters.
start	the starting offset in the character array.
length	the number of characters to use.

Returns

the new string.

Throws

IndexOutOfBoundsException	if `length < 0`, `start < 0` or `start + length > data.length`
NullPointerException	if `data` is `null`.

public static String **valueOf** (boolean value)

Added in API level 1

Converts the specified boolean to its string representation. When the boolean is `true` return `"true"`, otherwise return `"false"`.

Parameters

value	the boolean.

Returns

the boolean converted to a string.

public static String **valueOf** (char value)

Added in API level 1

Converts the specified character to its string representation.

Parameters

value	the character.

Returns

the character converted to a string.