|  |  |
| --- | --- |
| /\*\*  \* The {@code String} class represents character strings. All  \* string literals in Java programs, such as {@code "abc"}, are  \* implemented as instances of this class.  \* <p>  \* Strings are constant; their values cannot be changed after they  \* are created. String buffers support mutable strings.  \* Because String objects are immutable they can be shared. For example:  \* <blockquote><pre>  \* String str = "abc";  \* </pre></blockquote><p>  \* is equivalent to:  \* <blockquote><pre>  \* char data[] = {'a', 'b', 'c'};  \* String str = new String(data);  \* </pre></blockquote><p>  \* Here are some more examples of how strings can be used:  \* <blockquote><pre>  \* System.out.println("abc");  \* String cde = "cde";  \* System.out.println("abc" + cde);  \* String c = "abc".substring(2,3);  \* String d = cde.substring(1, 2);  \* </pre></blockquote>  \* <p>  \* The class {@code String} includes methods for examining  \* individual characters of the sequence, for comparing strings, for  \* searching strings, for extracting substrings, and for creating a  \* copy of a string with all characters translated to uppercase or to  \* lowercase. Case mapping is based on the Unicode Standard version  \* specified by the {@link java.lang.Character Character} class.  \* <p>  \* The Java language provides special support for the string  \* concatenation operator (&nbsp;+&nbsp;), and for conversion of  \* other objects to strings. String concatenation is implemented  \* through the {@code StringBuilder}(or {@code StringBuffer})  \* class and its {@code append} method.  \* String conversions are implemented through the method  \* {@code toString}, defined by {@code Object} and  \* inherited by all classes in Java. For additional information on  \* string concatenation and conversion, see Gosling, Joy, and Steele,  \* <i>The Java Language Specification</i>.  \*  \* <p> Unless otherwise noted, passing a <tt>null</tt> argument to a constructor  \* or method in this class will cause a {@link NullPointerException} to be  \* thrown.  \*  \* <p>A {@code String} represents a string in the UTF-16 format  \* in which <em>supplementary characters</em> are represented by <em>surrogate  \* pairs</em> (see the section <a href="Character.html#unicode">Unicode  \* Character Representations</a> in the {@code Character} class for  \* more information).  \* Index values refer to {@code char} code units, so a supplementary  \* character uses two positions in a {@code String}.  \* <p>The {@code String} class provides methods for dealing with  \* Unicode code points (i.e., characters), in addition to those for  \* dealing with Unicode code units (i.e., {@code char} values).  \*  \* @author Lee Boynton  \* @author Arthur van Hoff  \* @author Martin Buchholz  \* @author Ulf Zibis  \* @see java.lang.Object#toString()  \* @see java.lang.StringBuffer  \* @see java.lang.StringBuilder  \* @see java.nio.charset.Charset  \* @since JDK1.0  \*/ | compact1, compact2, compact3  java.lang Class String  * [java.lang.Object](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html)   + java.lang.String * All Implemented Interfaces:   [Serializable](https://docs.oracle.com/javase/8/docs/api/java/io/Serializable.html), [CharSequence](https://docs.oracle.com/javase/8/docs/api/java/lang/CharSequence.html), [Comparable](https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html)<[String](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html)>  public final class String  extends [Object](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html)  implements [Serializable](https://docs.oracle.com/javase/8/docs/api/java/io/Serializable.html), [Comparable](https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html)<[String](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html)>, [CharSequence](https://docs.oracle.com/javase/8/docs/api/java/lang/CharSequence.html)  The String class represents character strings. All string literals in Java programs, such as "abc", are implemented as instances of this class.  Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared. For example:  String str = "abc";    is equivalent to:  char data[] = {'a', 'b', 'c'};  String str = new String(data);    Here are some more examples of how strings can be used:  System.out.println("abc");  String cde = "cde";  System.out.println("abc" + cde);  String c = "abc".substring(2,3);  String d = cde.substring(1, 2);    The class String includes methods for examining individual characters of the sequence, for comparing strings, for searching strings, for extracting substrings, and for creating a copy of a string with all characters translated to uppercase or to lowercase. Case mapping is based on the Unicode Standard version specified by the [Character](https://docs.oracle.com/javase/8/docs/api/java/lang/Character.html) class.  The Java language provides special support for the string concatenation operator ( + ), and for conversion of other objects to strings. String concatenation is implemented through the StringBuilder(or StringBuffer) class and its append method. String conversions are implemented through the method toString, defined by Object and inherited by all classes in Java. For additional information on string concatenation and conversion, see Gosling, Joy, and Steele, *The Java Language Specification*.  Unless otherwise noted, passing a null argument to a constructor or method in this class will cause a [NullPointerException](https://docs.oracle.com/javase/8/docs/api/java/lang/NullPointerException.html) to be thrown.  A String represents a string in the UTF-16 format in which supplementary characters are represented by surrogate pairs (see the section [Unicode Character Representations](https://docs.oracle.com/javase/8/docs/api/java/lang/Character.html#unicode) in the Character class for more information). Index values refer to char code units, so a supplementary character uses two positions in a String.  The String class provides methods for dealing with Unicode code points (i.e., characters), in addition to those for dealing with Unicode code units (i.e., char values).  Since:  JDK1.0  See Also:  [Object.toString()](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html#toString--), [StringBuffer](https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuffer.html), [StringBuilder](https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html), [Charset](https://docs.oracle.com/javase/8/docs/api/java/nio/charset/Charset.html), [Serialized Form](https://docs.oracle.com/javase/8/docs/api/serialized-form.html#java.lang.String) |

|  |  |
| --- | --- |
| **Method Summary** | |
| char | [**charAt**](https://docs.oracle.com/javase/6/docs/api/java/lang/String.html#charAt%28int%29)(int index)            Returns the char value at the specified index. |

|  |
| --- |
| **Field Detail** |

**CASE\_INSENSITIVE\_ORDER**

public static final [Comparator](https://docs.oracle.com/javase/6/docs/api/java/util/Comparator.html)<[String](https://docs.oracle.com/javase/6/docs/api/java/lang/String.html)> **CASE\_INSENSITIVE\_ORDER**

A Comparator that orders String objects as by compareToIgnoreCase. This comparator is serializable.

Note that this Comparator does *not* take locale into account, and will result in an unsatisfactory ordering for certain locales. The java.text package provides *Collators* to allow locale-sensitive ordering.

**Since:**

1.2

**See Also:**

[Collator.compare(String, String)](https://docs.oracle.com/javase/6/docs/api/java/text/Collator.html#compare%28java.lang.String,%20java.lang.String%29)

|  |
| --- |
| **Constructor Detail** |

**String**

[@Deprecated](https://docs.oracle.com/javase/6/docs/api/java/lang/Deprecated.html)

public **String**(byte[] ascii,

int hibyte)

**Deprecated.** *This method does not properly convert bytes into characters. As of JDK 1.1, the preferred way to do this is via the String constructors that take a* [*Charset*](https://docs.oracle.com/javase/6/docs/api/java/nio/charset/Charset.html)*, charset name, or that use the platform's default charset.*

Allocates a new String containing characters constructed from an array of 8-bit integer values. Each character *c*in the resulting string is constructed from the corresponding component *b* in the byte array such that:

***c*** == (char)(((hibyte & 0xff) << 8)

| (***b*** & 0xff))

**Parameters:**

ascii - The bytes to be converted to characters

hibyte - The top 8 bits of each 16-bit Unicode code unit

**See Also:**

[String(byte[], int, int, java.lang.String)](https://docs.oracle.com/javase/6/docs/api/java/lang/String.html#String%28byte[],%20int,%20int,%20java.lang.String%29), [String(byte[], int, int, java.nio.charset.Charset)](https://docs.oracle.com/javase/6/docs/api/java/lang/String.html#String%28byte[],%20int,%20int,%20java.nio.charset.Charset%29), [String(byte[], int, int)](https://docs.oracle.com/javase/6/docs/api/java/lang/String.html#String%28byte[],%20int,%20int%29), [String(byte[], java.lang.String)](https://docs.oracle.com/javase/6/docs/api/java/lang/String.html#String%28byte[],%20java.lang.String%29), [String(byte[], java.nio.charset.Charset)](https://docs.oracle.com/javase/6/docs/api/java/lang/String.html#String%28byte[],%20java.nio.charset.Charset%29), [String(byte[])](https://docs.oracle.com/javase/6/docs/api/java/lang/String.html#String%28byte[]%29)

### String

public **String**(byte[] bytes,

int offset,

int length,

[String](https://docs.oracle.com/javase/6/docs/api/java/lang/String.html) charsetName)

throws [UnsupportedEncodingException](https://docs.oracle.com/javase/6/docs/api/java/io/UnsupportedEncodingException.html)

Constructs a new String by decoding the specified subarray of bytes using the specified charset. The length of the new String is a function of the charset, and hence may not be equal to the length of the subarray.

The behavior of this constructor when the given bytes are not valid in the given charset is unspecified. The [CharsetDecoder](https://docs.oracle.com/javase/6/docs/api/java/nio/charset/CharsetDecoder.html) class should be used when more control over the decoding process is required.

**Parameters:**

bytes - The bytes to be decoded into characters

offset - The index of the first byte to decode

length - The number of bytes to decode

charsetName - The name of a supported [charset](https://docs.oracle.com/javase/6/docs/api/java/nio/charset/Charset.html)

**Throws:**

[UnsupportedEncodingException](https://docs.oracle.com/javase/6/docs/api/java/io/UnsupportedEncodingException.html) - If the named charset is not supported

[IndexOutOfBoundsException](https://docs.oracle.com/javase/6/docs/api/java/lang/IndexOutOfBoundsException.html) - If the offset and length arguments index characters outside the bounds of the bytes array

**Since:**

JDK1.1

|  |
| --- |
| **Method Detail** |

**length**

public int **length**()

Returns the length of this string. The length is equal to the number of [Unicode code units](https://docs.oracle.com/javase/6/docs/api/java/lang/Character.html#unicode) in the string.

**Specified by:**

[length](https://docs.oracle.com/javase/6/docs/api/java/lang/CharSequence.html#length%28%29) in interface [CharSequence](https://docs.oracle.com/javase/6/docs/api/java/lang/CharSequence.html)

**Returns:**

the length of the sequence of characters represented by this object.

**codePointAt**

public int **codePointAt**(int index)

Returns the character (Unicode code point) at the specified index. The index refers to char values (Unicode code units) and ranges from 0 to [length()](https://docs.oracle.com/javase/6/docs/api/java/lang/String.html#length%28%29) - 1.

If the char value specified at the given index is in the high-surrogate range, the following index is less than the length of this String, and the char value at the following index is in the low-surrogate range, then the supplementary code point corresponding to this surrogate pair is returned. Otherwise, the char value at the given index is returned.

**Parameters:**

index - the index to the char values

**Returns:**

the code point value of the character at the index

**Throws:**

[IndexOutOfBoundsException](https://docs.oracle.com/javase/6/docs/api/java/lang/IndexOutOfBoundsException.html) - if the index argument is negative or not less than the length of this string.

**Since:**

1.5