Class java.lang.String

```
java.lang.Object
   |
   +----java.lang.String
```

public final class **String**
extends Object

A general class of objects to represent character Strings. Strings are constant, their values cannot be changed after creation. The compiler makes sure that each String constant actually results in a String object. 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);
```

See Also:
StringBuffer
Version:
1.51, 09/27/95
Author:
Lee Boynton, Arthur van Hoff

Constructor Index

o **String**()

Constructs a new empty String.

o **String**(String)
: Constructs a new String that is a copy of the specified String.

o **String**(char[])
: Constructs a new String whose initial value is the specified array of characters.

o **String**(char[], int, int)
: Constructs a new String whose initial value is the specified sub array of characters.

o **String**(byte[], int, int, int)
: Constructs a new String whose initial value is the specified sub array of bytes.

o **String**(byte[], int)
: Constructs a new String whose initial value is the specified array of bytes.

Method Index

o **charAt**(int)
: Returns the character at the specified index.

o **compareTo**(String)
: Compares this String to another specified String.

o **concat**(String)
: Concatenates the specified string to the end of this String.

o **copyValueOf**(char[], int, int)
: Returns a String that is equivalent to the specified character array.

o **copyValueOf**(char[])
: Returns a String that is equivalent to the specified character array.

o **endsWith**(String)
: Determines whether the String ends with some suffix.

o **equals**(Object)
: Compares this String to the specified object.

o **equalsIgnoreCase**(String)
: Compares this String to another object.

o **getBytes**(int, int, byte[], int)
: Copies characters from this String into the specified byte array.

o **getChars**(int, int, char[], int)
: Copies characters from this String into the specified character array.

o **hashCode**()
: Returns a hashcode for this String.

o **indexOf**(int)
: Returns the index within this String of the first occurrence of the specified character.

o **indexOf**(int, int)
: Returns the index within this String of the first occurrence of the specified character, starting the search at fromIndex.

o **indexOf**(String)
: Returns the index within this String of the first occurrence of the specified substring.

o **indexOf**(String, int)
: Returns the index within this String of the first occurrence of the specified

substring.

o **intern**()
Returns a String that is equal to this String but which is guaranteed to be from the unique String pool.

o **lastIndexOf**(int)
Returns the index within this String of the last occurrence of the specified character.

o **lastIndexOf**(int, int)
Returns the index within this String of the last occurrence of the specified character.

o **lastIndexOf**(String)
Returns the index within this String of the last occurrence of the specified substring.

o **lastIndexOf**(String, int)
Returns the index within this String of the last occurrence of the specified substring.

o **length**()
Returns the length of the String.

o **regionMatches**(int, String, int, int)
Determines whether a region of this String matches the specified region of the specified String.

o **regionMatches**(boolean, int, String, int, int)
Determines whether a region of this String matches the specified region of the specified String.

o **replace**(char, char)
Converts this String by replacing all occurences of oldChar with newChar.

o **startsWith**(String, int)
Determines whether this String starts with some prefix.

o **startsWith**(String)
Determines whether this String starts with some prefix.

o **substring**(int)
Returns the substring of this String.

o **substring**(int, int)
Returns the substring of a String.

o **toCharArray**()
Converts this String to a character array.

o **toLowerCase**()
Converts all of the characters in this String to lower case.

o **toString**()
Converts this String to a String.

o **toUpperCase**()
Converts all of the characters in this String to upper case.

o **trim**()
Trims leading and trailing whitespace from this String.

o **valueOf**(Object)
Returns a String that represents the String value of the object.

o **valueOf**(char[])
Returns a String that is equivalent to the specified character array.

o **valueOf**(char[], int, int)

Returns a String that is equivalent to the specified character array.

o **valueOf**(boolean)
  Returns a String object that represents the state of the specified boolean.

o **valueOf**(char)
  Returns a String object that contains a single character

o **valueOf**(int)
  Returns a String object that represents the value of the specified integer.

o **valueOf**(long)
  Returns a String object that represents the value of the specified long.

o **valueOf**(float)
  Returns a String object that represents the value of the specified float.

o **valueOf**(double)
  Returns a String object that represents the value of the specified double.

Constructors

o **String**

```
public String()
```

Constructs a new empty String.

o **String**

```
public String(String value)
```

Constructs a new String that is a copy of the specified String.
Parameters:
value – the initial value of the String

o **String**

```
public String(char value[])
```

Constructs a new String whose initial value is the specified array of characters. The character array is **NOT** copied, so **DO NOT** modify the array after the String is created!
Parameters:
value – the initial value of the String

o **String**

```
public String(char value[],
              int offset,
              int count)
```

Constructs a new String whose initial value is the specified sub array of characters. The length of the new string will be count characters starting at offset within the specified character array. The character array is **NOT** copied, so **DO NOT** modify the array after the String is created!

Parameters:

value – the initial value of the String, an array of characters
offset – the offset into the value of the String
count – the length of the value of the String

Throws: StringIndexOutOfBoundsException

If the offset and count arguments are invalid.

o **String**

```
public String(byte ascii[],
              int hibyte,
              int offset,
              int count)
```

Constructs a new String whose initial value is the specified sub array of bytes. The high–byte of each character can be specified, it should usually be 0. The length of the new String will be count characters starting at offset within the specified character array.

Parameters:

ascii – the bytes that will be converted to characters
hibyte – the high byte of each Unicode character
offset – the offset into the ascii array
count – the length of the String

Throws: StringIndexOutOfBoundsException

If the offset and count arguments are invalid.

o **String**

```
public String(byte ascii[],
              int hibyte)
```

Constructs a new String whose initial value is the specified array of bytes. The byte array transformed into Unicode chars using hibyte as the upper byte of each character.

Parameters:

ascii – the byte that will be converted to characters
hibyte – the top 8 bits of each 16 bit Unicode character

Methods

o **length**

```
public int length()
```

Returns the length of the String. The length of the String is equal to the number of 16 bit Unicode characters in the String.

o **charAt**

```
public char charAt(int index)
```

Returns the character at the specified index. An index ranges from `0` to `length() - 1`.
Parameters:
  index – the index of the desired character
Throws: StringIndexOutOfBoundsException
  If the index is not in the range `0` to `length()-1`.

o **getChars**

```
public void getChars(int srcBegin,
                     int srcEnd,
                     char dst[],
                     int dstBegin)
```

Copies characters from this String into the specified character array. The characters of the specified substring (determined by srcBegin and srcEnd) are copied into the character array, starting at the array's dstBegin location.
Parameters:
  srcBegin – index of the first character in the string
  srcEnd – end of the characters that are copied
  dst – the destination array
  dstBegin – the start offset in the destination array

o **getBytes**

```
public void getBytes(int srcBegin,
                     int srcEnd,
                     byte dst[],
                     int dstBegin)
```

Copies characters from this String into the specified byte array. Copies the characters of the specified substring (determined by srcBegin and srcEnd) into the byte array, starting at the array's dstBegin location.
Parameters:
  srcBegin – index of the first character in the String
  srcEnd – end of the characters that are copied
  dst – the destination array
  dstBegin – the start offset in the destination array

o **equals**

```
public boolean equals(Object anObject)
```

Compares this String to the specified object. Returns true if the object is equal to this String; that is, has the same length and the same characters in the same sequence.

Parameters:
anObject – the object to compare this String against
Returns:
true if the Strings are equal; false otherwise.
Overrides:
equals in class Object

o **equalsIgnoreCase**

```
public boolean equalsIgnoreCase(String anotherString)
```

Compares this String to another object. Returns true if the object is equal to this String; that is, has the same length and the same characters in the same sequence. Upper case characters are folded to lower case before they are compared.
Parameters:
anotherString – the String to compare this String against
Returns:
true if the Strings are equal, ignoring case; false otherwise.

o **compareTo**

```
public int compareTo(String anotherString)
```

Compares this String to another specified String. Returns an integer that is less than, equal to, or greater than zero. The integer's value depends on whether this String is less than, equal to, or greater than anotherString.
Parameters:
anotherString – the String to be compared

o **regionMatches**

```
public boolean regionMatches(int toffset,
                             String other,
                             int ooffset,
                             int len)
```

Determines whether a region of this String matches the specified region of the specified String.
Parameters:
toffset – where to start looking in this String
other – the other String
ooffset – where to start looking in the other String
len – the number of characters to compare
Returns:
true if the region matches with the other; false otherwise.

o **regionMatches**

```
public boolean regionMatches(boolean ignoreCase,
                             int toffset,
                                String other,
                                int ooffset,
                                int len)
```

Determines whether a region of this String matches the specified region of the specified String. If the boolean ignoreCase is true, upper case characters are considered equivalent to lower case letters.

Parameters:

ignoreCase – if true, case is ignored
toffset – where to start looking in this String
other – the other String
ooffset – where to start looking in the other String
len – the number of characters to compare

Returns:

true if the region matches with the other; false otherwise.

o **startsWith**

```
public boolean startsWith(String prefix,
                          int toffset)
```

Determines whether this String starts with some prefix.

Parameters:

prefix – the prefix
toffset – where to begin looking in the the String

Returns:

true if the String starts with the specified prefix; false otherwise.

o **startsWith**

```
public boolean startsWith(String prefix)
```

Determines whether this String starts with some prefix.

Parameters:

prefix – the prefix

Returns:

true if the String starts with the specified prefix; false otherwise.

o **endsWith**

```
public boolean endsWith(String suffix)
```

Determines whether the String ends with some suffix.

Parameters:

suffix – the suffix

Returns:

true if the String ends with the specified suffix; false otherwise.

o **hashCode**

```
public int hashCode()
```

Returns a hashcode for this String. This is a large number composed of the character values in the String.
Overrides:
hashCode in class Object

o **indexOf**

```
public int indexOf(int ch)
```

Returns the index within this String of the first occurrence of the specified character. This method returns –1 if the index is not found.
Parameters:
ch – the character to search for

o **indexOf**

```
public int indexOf(int ch,
                   int fromIndex)
```

Returns the index within this String of the first occurrence of the specified character, starting the search at fromIndex. This method returns –1 if the index is not found.
Parameters:
ch – the character to search for
fromIndex – the index to start the search from

o **lastIndexOf**

```
public int lastIndexOf(int ch)
```

Returns the index within this String of the last occurrence of the specified character. The String is searched backwards starting at the last character. This method returns –1 if the index is not found.
Parameters:
ch – the character to search for

o **lastIndexOf**

```
public int lastIndexOf(int ch,
                       int fromIndex)
```

Returns the index within this String of the last occurrence of the specified character. The String is searched backwards starting at fromIndex. This method returns –1 if the index is not found.
Parameters:
ch – the character to search for

fromIndex – the index to start the search from

o **indexOf**

```
public int indexOf(String str)
```

Returns the index within this String of the first occurrence of the specified substring. This method returns –1 if the index is not found.
Parameters:
str – the substring to search for

o **indexOf**

```
public int indexOf(String str,
                   int fromIndex)
```

Returns the index within this String of the first occurrence of the specified substring. The search is started at fromIndex. This method returns –1 if the index is not found.
Parameters:
str – the substring to search for
fromIndex – the index to start the search from

o **lastIndexOf**

```
public int lastIndexOf(String str)
```

Returns the index within this String of the last occurrence of the specified substring. The String is searched backwards. This method returns –1 if the index is not found.
Parameters:
str – the substring to search for

o **lastIndexOf**

```
public int lastIndexOf(String str,
                       int fromIndex)
```

Returns the index within this String of the last occurrence of the specified substring. The String is searched backwards starting at fromIndex. This method returns –1 if the index is not found.
Parameters:
str – the substring to search for
fromIndex – the index to start the search from

o **substring**

```
public String substring(int beginIndex)
```

Returns the substring of this String. The substring is specified by a beginIndex (inclusive) and the end of the string.
Parameters:
beginIndex – the beginning index, inclusive

o **substring**

```
public String substring(int beginIndex,
                        int endIndex)
```

Returns the substring of a String. The substring is specified by a beginIndex (inclusive) and an endIndex (exclusive).
Parameters:
beginIndex – the beginning index, inclusive
endIndex – the ending index, exclusive
Throws: StringIndexOutOfBoundsException
If the beginIndex or the endIndex is out of range.

o **concat**

```
public String concat(String str)
```

Concatenates the specified string to the end of this String.
Parameters:
str – the String which is concatenated to the end of this String

o **replace**

```
public String replace(char oldChar,
                      char newChar)
```

Converts this String by replacing all occurences of oldChar with newChar.
Parameters:
oldChar – the old character
newChar – the new character

o **toLowerCase**

```
public String toLowerCase()
```

Converts all of the characters in this String to lower case.
Returns:
the String, converted to lowercase.
See Also:
toLowerCase, toUpperCase

o **toUpperCase**

```
public String toUpperCase()
```

Converts all of the characters in this String to upper case.
Returns:
the String, converted to uppercase.
See Also:
toUpperCase, toLowerCase

o **trim**

```
public String trim()
```

Trims leading and trailing whitespace from this String.
Returns:
the String, with whitespace removed.

o **toString**

```
public String toString()
```

Converts this String to a String.
Returns:
the String itself.
Overrides:
toString in class Object

o **toCharArray**

```
public char[] toCharArray()
```

Converts this String to a character array. This creates a new array.
Returns:
an array of characters.

o **valueOf**

```
public static String valueOf(Object obj)
```

Returns a String that represents the String value of the object. The object may choose how to represent itself by implementing the toString() method.
Parameters:
obj – the object to be converted

o **valueOf**

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

Returns a String that is equivalent to the specified character array. Uses the original array as the body of the String (ie. it does not copy it to a new array).
Parameters:
data – the character array

o **valueOf**

```
public static String valueOf(char data[],
                             int offset,
                             int count)
```

Returns a String that is equivalent to the specified character array. Uses the original array as the body of the String (ie. it does not copy it to a new array).
Parameters:
data – the character array
offset – the offset into the value of the String
count – the length of the value of the String

o **copyValueOf**

```
public static String copyValueOf(char data[],
                                 int offset,
                                 int count)
```

Returns a String that is equivalent to the specified character array. It creates a new array and copies the characters into it.
Parameters:
data – the character array
offset – the offset into the value of the String
count – the length of the value of the String

o **copyValueOf**

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

Returns a String that is equivalent to the specified character array. It creates a new array and copies the characters into it.
Parameters:
data – the character array

o **valueOf**

```
public static String valueOf(boolean b)
```

Returns a String object that represents the state of the specified boolean.
Parameters:
b – the boolean

o **valueOf**

```
public static String valueOf(char c)
```

Returns a String object that contains a single character
Parameters:
c – the character

Returns:
the resulting String.

o **valueOf**

```
public static String valueOf(int i)
```

Returns a String object that represents the value of the specified integer.
Parameters:
i – the integer

o **valueOf**

```
public static String valueOf(long l)
```

Returns a String object that represents the value of the specified long.
Parameters:
l – the long

o **valueOf**

```
public static String valueOf(float f)
```

Returns a String object that represents the value of the specified float.
Parameters:
f – the float

o **valueOf**

```
public static String valueOf(double d)
```

Returns a String object that represents the value of the specified double.
Parameters:
d – the double

o **intern**

```
public String intern()
```

Returns a String that is equal to this String but which is guaranteed to be from the unique String pool. For example:

```
s1.intern() == s2.intern() <=> s1.equals(s2).
```
