

Java StringBuffer Class Tutorial

The `StringBuffer` Class

Java provides two classes that can be used for storing and manipulating character strings. The `String` class, introduced earlier, is used for constant strings (i.e., strings whose characters won't be changed). The `StringBuffer` class, introduced here, is used for strings that can change during program execution.

Some of the frequently used methods of the `StringBuffer` class are specified in the table below.

Method	Description
<code>append(arg)</code>	Appends the value stored in <code>arg</code> to the end of a <code>StringBuffer</code> object. <code>arg</code> can be any primitive type, or a <code>String</code> type. The buffer size will be increased if necessary.
<code>append(String s, int start, int num)</code>	Appends a portion of string <code>s</code> to a <code>StringBuffer</code> object, starting at index position <code>start</code> , and appends <code>num</code> characters. Buffer size is increased if necessary.
<code>int capacity()</code>	Returns the capacity of the <code>StringBuffer</code> object.
<code>char charAt(int index)</code>	Returns the character at location specified by <code>index</code> .
<code>ensureCapacity(int size)</code>	Changes the default capacity of the buffer to <code>size</code> characters.
<code>boolean equals(Object o)</code>	Returns true if the string equals <code>o</code> .
<code>insert(int pos, arg)</code>	Inserts the string value of <code>arg</code> at index position <code>pos</code> of the <code>StringBuffer</code> object. <code>arg</code> can be any primitive type or a <code>String</code> type.
<code>int length()</code>	Returns the length of the string as an integer.
<code>reverse()</code>	Reverses the characters in a <code>StringBuffer</code> object.

Java StringBuffer Class Tutorial

<code>setChar(index pos, char c)</code>	Sets the character at index <code>pos</code> to the value <code>c</code> .
<code>setLength(int num)</code>	Sets the length of the string of the <code>StringBuffer</code> object to <code>num</code> characters. Extra characters are set to the null character (ASCII <code>'\0'</code> or unicode <code>'\u0000'</code>). If <code>num</code> is less than the length of the current string, characters will be truncated.
<code>String substring(int bdx)</code>	Returns a new string consisting of all the characters from the beginning index, <code>bdx</code> , until the end of the string.
<code>String substring(int bdx, int edx)</code>	Returns a new string consisting of all the characters from the beginning index, <code>bdx</code> , until the ending index (exclusive) <code>edx</code> .

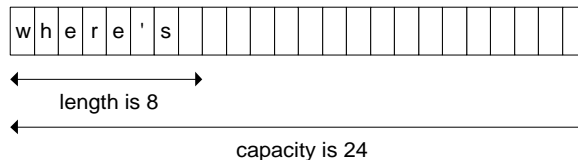
Creating a StringBuffer Object

There are three (3) ways to create a `StringBuffer` object, as illustrated below:

```
StringBuffer objectName = new StringBuffer( );  
  
StringBuffer objectName = new StringBuffer( int size );  
  
StringBuffer objectName = new StringBuffer( String s );
```

The first statement above creates a `StringBuffer` object with no characters in it and with a buffer capacity of 16 characters. The second statement creates a `StringBuffer` object with no characters and a buffer capacity of `size` characters. The third statement creates a `StringBuffer` object that is initialized to the characters contained in the string `s` and has an initial buffer size of 16 characters plus the length of `s`.

```
StringBuffer buf = new StringBuffer( "where's " );
```



It is important to note that a buffer is a block of memory that Java allocates to hold a `StringBuffer` object. The length of a particular `StringBuffer` object can be different than the length of the buffer. The length of the buffer is called the **buffer capacity**. When you create a `StringBuffer` object from an existing `String` object, as illustrated above, the buffer capacity defaults to the length of the `String` object plus 16 characters. The default capacity can be changed using the `ensureCapacity()` method.

Java StringBuffer Class Tutorial

Manipulating StringBuffer Objects

There are a number of `StringBuffer` methods that can be used to manipulate `StringBuffer` objects. The most commonly used are `append()`, `insert()`, and `setChar()`.

When characters are added to a `StringBuffer` object Java automatically increases the buffer capacity if necessary. The buffer is increased if the length of the current `StringBuffer` object plus the length of the added characters exceeds the existing buffer capacity. The current buffer capacity is returned by the `capacity()` method.

```
String s = "mini me?";
```

m	i	n	i		m	e	?
---	---	---	---	--	---	---	---

```
buf.append( s );
```

[illegible]

length is 16

capacity is 24

In the above example, the `append()` method is used to add characters to the existing `StringBuffer` object `buf`. Since the current length of `buf` is eight (8) and the length of the string being added is eight (8) the buffer capacity is not changed.

The program below demonstrates some of the other `StringBuffer` methods.

Java StringBuffer Class Tutorial

```
public class StringBufferDemo
{
    public static void main( String [] args )
    {
        String s1 = "Groovy Baby";
        String s2 = "mini me?";
        String s3 = "Four score and seven years ago";
        String bc = "Buffer capacity = ";

        StringBuffer s = new StringBuffer("Where's ");

        // Display length of string & buffer

        System.out.println( "\n" + "String is " + s );
        System.out.println( bc + s.capacity() );
        System.out.println( "String length    = " +
                             s.length() );
        System.out.println();

        // Append string

        s.append( s2 );
        System.out.println( "String is " + s );
        System.out.println( bc + s.capacity() );
        System.out.println();

        // Trim string & insert characters

        s.setLength( 0 );
        s.append( s1 );
        System.out.println( "String is " + s );
        System.out.println( bc + s.capacity() );
        System.out.println();

        // Insert string

        s.insert( 11, s3 );
        System.out.println( "String is " + s );
        System.out.println( bc + s.capacity() );
        System.out.println();

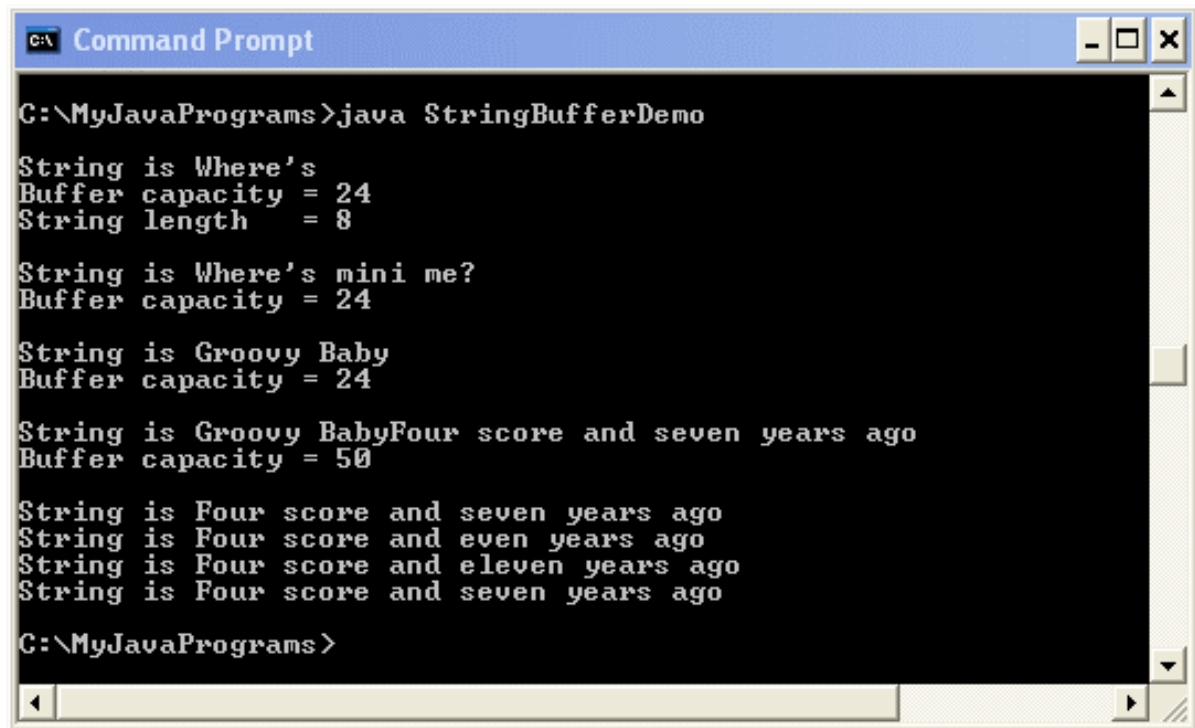
        // Trim string, delete & add

        s.setLength( 0 );
        s.append( s3 );
        System.out.println( "String is " + s );
        s.deleteCharAt( 15 );
        System.out.println( "String is " + s );
        s.insert( 15, "el" );
        System.out.println( "String is " + s );

        s.delete( 15,16 );    // Delete 'e' in 'eleven'
        s.setCharAt( 15,'s' );
        System.out.println( "String is " + s );
    }
}
```

Java StringBuffer Class Tutorial

The program output is as follows:



```
C:\> Command Prompt

C:\MyJavaPrograms>java StringBufferDemo

String is Where's
Buffer capacity = 24
String length    = 8

String is Where's mini me?
Buffer capacity = 24

String is Groovy Baby
Buffer capacity = 24

String is Groovy BabyFour score and seven years ago
Buffer capacity = 50

String is Four score and seven years ago
String is Four score and even years ago
String is Four score and eleven years ago
String is Four score and seven years ago

C:\MyJavaPrograms>
```