**StringBuffer**

StringBuffer sentence = new StringBuffer();

sentence.append(w);

sentence.toString();

<http://docs.oracle.com/javase/7/docs/api/java/lang/StringBuffer.html>

java.lang

## Class StringBuffer

* [java.lang.Object](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)
  + java.lang.StringBuffer
* **All Implemented Interfaces:**

[Serializable](http://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html), [Appendable](http://docs.oracle.com/javase/7/docs/api/java/lang/Appendable.html" \o "interface in java.lang), [CharSequence](http://docs.oracle.com/javase/7/docs/api/java/lang/CharSequence.html" \o "interface in java.lang)

public final class ****StringBuffer****

extends [Object](http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html)

implements [Serializable](http://docs.oracle.com/javase/7/docs/api/java/io/Serializable.html), [CharSequence](http://docs.oracle.com/javase/7/docs/api/java/lang/CharSequence.html)

A thread-safe, mutable sequence of characters. A string buffer is like a [String](http://docs.oracle.com/javase/7/docs/api/java/lang/String.html), but can be modified. At any point in time it contains some particular sequence of characters, but the length and content of the sequence can be changed through certain method calls.

String buffers are safe for use by multiple threads. The methods are synchronized where necessary so that all the operations on any particular instance behave as if they occur in some serial order that is consistent with the order of the method calls made by each of the individual threads involved.

The principal operations on a StringBuffer are the append and insert methods, which are overloaded so as to accept data of any type. Each effectively converts a given datum to a string and then appends or inserts the characters of that string to the string buffer. The append method always adds these characters at the end of the buffer; the insert method adds the characters at a specified point.

For example, if z refers to a string buffer object whose current contents are "start", then the method call z.append("le") would cause the string buffer to contain "startle", whereas z.insert(4, "le") would alter the string buffer to contain "starlet".

In general, if sb refers to an instance of a StringBuffer, then sb.append(x) has the same effect as sb.insert(sb.length(), x).

Whenever an operation occurs involving a source sequence (such as appending or inserting from a source sequence) this class synchronizes only on the string buffer performing the operation, not on the source.

Every string buffer has a capacity. As long as the length of the character sequence contained in the string buffer does not exceed the capacity, it is not necessary to allocate a new internal buffer array. If the internal buffer overflows, it is automatically made larger. As of release JDK 5, this class has been supplemented with an equivalent class designed for use by a single thread, [StringBuilder](http://docs.oracle.com/javase/7/docs/api/java/lang/StringBuilder.html" \o "class in java.lang). The StringBuilder class should generally be used in preference to this one, as it supports all of the same operations but it is faster, as it performs no synchronization.

****Since:****

JDK1.0

****See Also:****

[StringBuilder](http://docs.oracle.com/javase/7/docs/api/java/lang/StringBuilder.html), [String](http://docs.oracle.com/javase/7/docs/api/java/lang/String.html), [Serialized Form](http://docs.oracle.com/javase/7/docs/api/serialized-form.html#java.lang.StringBuffer)