public final class StringBuffer

extends Object
implements Serializable, CharSequence

A thread-safe, mutable sequence of characters. A string buffer is like a String, 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. Note that while `StringBuffer` is designed to be safe to use concurrently from multiple threads, if the constructor or the `append` or `insert` operation is passed a source sequence that is shared across threads, the calling code must ensure that the operation has a consistent and unchanging view of the source sequence for the duration of the operation. This could be satisfied by the caller holding a lock during the operation's call, by using an immutable source sequence, or by not sharing the source sequence across threads.

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.

Unless otherwise noted, passing a `null` argument to a constructor or method in this class will cause a NullPointerException to be thrown.

As of release JDK 5, this class has been supplemented with an equivalent class designed for use by a single thread, StringBuilder. 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.

public final class StringBuilder

extends Object
implements Serializable, CharSequence

A mutable sequence of characters. This class provides an API compatible with StringBuffer, but with no guarantee of synchronization. This class is designed for use as a drop-in replacement for

StringBuffer in places where the string buffer was being used by a single thread (as is generally the case). Where possible, it is recommended that this class be used in preference to StringBuffer as it will be faster under most implementations.

The principal operations on a `StringBuilder` 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 builder. The `append` method always adds these characters at the end of the builder; the `insert` method adds the characters at a specified point.

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

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

Every string builder has a capacity. As long as the length of the character sequence contained in the string builder does not exceed the capacity, it is not necessary to allocate a new internal buffer. If the internal buffer overflows, it is automatically made larger.

Instances of `StringBuilder` are not safe for use by multiple threads. If such synchronization is required then it is recommended that StringBuffer be used.

Unless otherwise noted, passing a `null` argument to a constructor or method in this class will cause a NullPointerException to be thrown.