# StringBuffer class in Java

StringBuffer class is used to create a mutable string object. It means, it can be changed after it is created. It represents growable and writable character sequence.

It is similar to String class in Java both are used to create string, but stringbuffer object can be changed.

So StringBuffer class is used when we have to make lot of modifications to our string. It is also thread safe i.e multiple threads cannot access it simultaneously. StringBuffer defines 4 constructors.

StringBuffer(): It creates an empty string buffer and reserves space for 16 characters.

StringBuffer(int size): It creates an empty string and takes an integer argument to set capacity of the buffer.

StringBuffer(String str): It creates a stringbuffer object from the specified string.

StringBuffer(charSequence []ch): It creates a stringbuffer object from the charsequence array.

Example: Creating a StringBuffer Object

In this example, we are creating string buffer object using StrigBuffer class and also testing its mutability.

```
public class Demo {

        public static void main(String[] args) {

                StringBuffer sb = new StringBuffer("study");

                System.out.println(sb);

                // modifying object

                sb.append("tonight");

                System.out.println(sb);    // Output: studytonight

        }

}
```

Output

```
study
studytonight
```

Example: difference between String and StringBuffer

In this example, we are creating objects of String and StringBuffer class and modifying them, but only stringbuffer object get modified. See the below example.

```
class Test {
 public static void main(String args[])
 {
  String str = "study";
  str.concat("tonight");
  System.out.println(str);      // Output: study


  StringBuffer strB = new StringBuffer("study");
  strB.append("tonight");
  System.out.println(strB);    // Output: studytonight
 }
}
```

Output

```
study
studytonight
```

Explanation:

Output is such because String objects are immutable objects. Hence, if we concatenate on the same String object, it won't be altered But StringBuffer creates mutable objects. Hence, it can be altered.

- **Important methods of StringBuffer class**

The following methods are some most commonly used methods of StringBuffer class.

1) append()

This method will concatenate the string representation of any type of data to the end of the StringBuffer object. append() method has several overloaded forms.

StringBuffer append(String str)

StringBuffer append(int n)

StringBuffer append(Object obj)

The string representation of each parameter is appended to StringBuffer object.

```
public class Demo {
	public static void main(String[] args) {
		StringBuffer str = new StringBuffer("test");
		str.append(123);
		System.out.println(str);
	}
}
```

O/P test123

2) insert()

This method inserts one string into another. Here are few forms of insert() method.

StringBuffer insert(int index, String str)

StringBuffer insert(int index, int num)

StringBuffer insert(int index, Object obj)

Here the first parameter gives the index at which position the string will be inserted and string representation of second parameter is inserted into StringBuffer object.

```
public class Demo {

        public static void main(String[] args) {

                StringBuffer str = new StringBuffer("test");

                str.insert(2, 123);

                System.out.println(str);

        }

}
```

O/P test123

3) reverse()

This method reverses the characters within a StringBuffer object.

```
public class Demo {

        public static void main(String[] args) {

                StringBuffer str = new StringBuffer("Hello");

                str.reverse();

                System.out.println(str);

        }

}
```

O/P olleH

4) replace()

This method replaces the string from specified start index to the end index.

```
public class Demo {

        public static void main(String[] args) {
                StringBuffer str = new StringBuffer("Hello World");
                str.replace( 6, 11, "java");
                System.out.println(str);
        }
}
```

O/P Hello java

5) capacity()

This method returns the current capacity of StringBuffer object.

```
public class Demo {
        public static void main(String[] args) {
                StringBuffer str = new StringBuffer();
                System.out.println( str.capacity() );
        }
}
```

O/P 16

Note: Empty constructor reserves space for 16 characters. Therefore the output is 16.

6) ensureCapacity()

This method is used to ensure minimum capacity of StringBuffer object.

If the argument of the ensureCapacity() method is less than the existing capacity, then there will be no change in existing capacity.

If the argument of the ensureCapacity() method is greater than the existing capacity, then there will be change in the current capacity using following rule: newCapacity = (oldCapacity*2) + 2.

```
public class Demo {
        public static void main(String[] args) {

                StringBuffer str = new StringBuffer();

                System.out.println( str.capacity()); //output: 16 (since empty
constructor reserves space for 16 characters)

                str.ensureCapacity(30); //greater than the existing capacity

                System.out.println( str.capacity()); //output: 34 (by following the
rule - (oldcapacity*2) + 2.) i.e (16*2)+2 = 34.

        }

}
```

Output

```
16

34
```