What is String Buffer?

- StringBuffer is mutable String.
- Java StringBuffer class is (synchronized)thread-safe i.e. multiple threads cannot access it simultaneously.
- So it is safe and will result in an order.

Why String Buffer is mutable?

```
public class StringDemo4 {

    public static void main(String[] args) {
    StringBuffer sb=new StringBuffer("Ajith ");
    sb.append("Kumar");
    System.out.println(sb);
    }

}
```

```
Ajith Kumar
```

Overloading

append() it return many data types

Methods

- append(), replace(), setCharAt()
- insert()
- delete()
- reverse()
- length()
- charAt()
- deleteChatAt()
- setLength()

Methods Name	Introduced Version	Description	Return Type	Parameter	E
delete()	1.2	Removes the characters in a substring of this sequence.	This object.	start - The beginning index, inclusive.end - The ending index, exclusive.	S B
deleteCharAt()	1.2	Removes the char at the specified position in this sequence. This sequence is shortened by one char.	“	Index of char to remove	
replace()	1.2	Replaces the characters in a substring of this sequence with characters in the specified String.	“	start - The beginning index, inclusive.end - The ending index, exclusive.str - String that will replace previous contents.	

Methods Name	Introduced Version	Description	Return Type	Parameter	E
reverse()	1.0.2	Causes this character sequence to be replaced by the reverse of the sequence	A reference to this object.	No	N
setLength()	1.0	Sets the length of the character sequence. The sequence is changed to a new character sequence whose length is specified by the argument	No	The new length	I c
charAt()	1.0	Returns the char value in this sequence at the specified index.	the char value at the specified index	index - the index of the desired char value.	I c
setCharAt()	1.0	The character at the specified index is set.	No	index - the index of the character to modify.ch - the new character.	I c

Methods Name	Introduced Version	Description(Overloading)	Return Type	Parameter	F
append()	1.0	Appends the specified string to this character sequence.	A reference to this object.	A string.	N
insert()	1.0	Inserts the string into this character sequence	A reference to this object.	offset - the offset.str - a string.	S B

Append() and insert() methods has many types in StringBuffer.

Example 1: delete(), deleteCharAt()

```
public class StringBufferDelete {

    public static void main(String[] args) {
StringBuffer sb=new StringBuffer("Java Programming is awesome");
System.out.println(sb.delete(5,10));
StringBuffer sb1=new StringBuffer("Python is super");
System.out.println(sb1.deleteCharAt(10));
    }

}
```

```
Java amming is awesome
Python is uper
```

Example 2: reverse(), replace()

```
public class StringBufferReplace {

    public static void main(String[] args) {
        StringBuffer sb=new StringBuffer("Java Programming is awesome");
        System.out.println(sb.replace(5, 15, "Python"));
        System.out.println(sb.reverse());
    }

}
```

```
Java Pythong is awesome
emosewa si gnohtyP avaJ
```

Example 3: setLength()

```
public class StringBufferTrim {

    public static void main(String[] args) {
        StringBuffer sb=new StringBuffer("Java Programming is awesome");
        sb.setLength(8);
        System.out.println(sb);
    }

}
```

Example 4:charAt(), setCharAt()

```
public class StringBufferChar {

    public static void main(String[] args) {
        StringBuffer sb=new StringBuffer("Java Programming is awesome");
        System.out.println(sb.charAt(5));
        sb.setCharAt(10, 'p');
        System.out.println(sb);
    }
}
```

```
P
Java Progrpmming is awesome
```

Example 5: insert()

```
public class StringBufferInsert {

    public static void main(String[] args) {
        StringBuffer sb=new StringBuffer("Java Programming is awesome");
        sb.insert(17, "nice");
        System.out.println(sb);
    }
```

Overloading

```
Java Programming niceis awesome
```

What is String Builder?

- StringBuilder is mutable String.
- The Java StringBuilder class is same as StringBuffer class except that it i non-synchronized (not thread-safe).
- It is available since JDK 1.5.

Methods

- append(), replace(), subsequence(), substring(), charAt() , trimToSize()
- insert() delete(), capacity(), ensureCapacity(), reverse(), length()
- StringBuffer methods is similar to StringBuilder

Methods Name	Introduced Version	Description(Overloading)	Return Type	Parameter	E
subsequence ()	1.5	the new character sequence from given start index to exclusive end index value of this sequence.	The subSequence(int start, int end) method returns the specified subsequence.	The start and end index of a subsequence.	In
trimToSize( )	1.5	To reduce the storage used for the character sequence.	No	No	N
reverse()	1.5	Causes this character sequence to be replaced by the reverse of the sequence	A reference to this object.	No	N

Example 1:reverse()

```
public class StringDemo6 {

    public static void main(String[] args) {
        StringBuilder sb=new StringBuilder("Hello");
        sb.reverse();
        System.out.println(sb);
    }

}
```

```
olleH
```

Overrides in a String Buffer

Example 2: subsequence()

```
public class StringBuilderSubSeq {

    public static void main(String[] args) {
        StringBuilder sb=new StringBuilder("Java is high level language");
        CharSequence cs=sb.subSequence(2, 10);
        System.out.println(cs);
```

```
va is hi
```

Example 3: trimToSize()

```
public class StringBuilderTrim {
    public static void main(String[] args) {
        StringBuilder sb = new StringBuilder("programming");
        System.out.println("String = "+sb);
        int length = sb.length();
        int capacity = sb.capacity();
        System.out.println("Length = "+length);
        System.out.println("Capacity = "+capacity);
        sb.trimToSize();
        length = sb.length();
        capacity = sb.capacity();
        System.out.println("Length after trimtosize = "+length);
        System.out.println("Capacity after trimtosize= "+capacity);
    }
}
```

```
String = programmi
Length = 11
Capacity = 27
Length after trimt
Capacity after tri
```

Difference

String	String Buffer
It is immutable	It is mutable.
String is slow and consumes more memory	String is fast and consumes less memory
When you concat too many strings because every time it creates new instance.	When you cancat strings.
1.0	1.0
Java.lang.package	Java.lang.pakage

Difference

StringBuffer	String Builder
It is immutable	It is immutable.
It is synchronized i.e. thread safe.	It is non-synchronized i.e. not thread safe.
It means two threads can't call the methods of StringBuffer simultaneously.	It means two threads can call the methods o StringBuilder simultaneously.
1.0	1.5
It is less efficient.	It is more efficient.