1 | **Java StringBuffer Class**

Java StringBuffer class is used to create mutable (modifiable) string object. A string buffer is like a String, but can be modified. As String objects are immutable, so if we do a lot of modifications to String objects, we may end up with a memory leak. To overcome this we use StringBuffer class.

StringBuffer class represents growable and writable character sequence. It is also thread-safe i.e. multiple threads cannot access it simultaneously. 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.

Constructors of StringBuffer Class –

- **StringBuffer( )-** Creates an empty string buffer with the initial capacity of 16.
- **StringBuffer(int capacity)-** Creates an empty string buffer with the specified capacity as length.
- **StringBuffer(String str)-** Creates a string buffer initialized to the contents of the specified string.
- **StringBuffer(charSequence[] ch)-** Creates a string buffer that contains the same characters as the specified CharSequence.

Important methods of StringBuffer class –

- **append()**

The append() method concatenates the given argument(string representation) to the end of the invoking StringBuffer object. StringBuffer class has several overloaded append() method.

- StringBuffer append(String str)
- StringBuffer append(int n)
- StringBuffer append(Object obj)

Example:

```
StringBuffer strBuffer = new StringBuffer("Hello");
strBuffer.append("World");
System.out.println(strBuffer);
strBuffer.append(101);
System.out.println(strBuffer);
```

Output:

```
Hello World
Hello World 101
```

- **insert()**

The insert() method inserts the given argument(string representation) into the invoking StringBuffer object at the given position. 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.

Example:

```
StringBuffer str = new StringBuffer("Test");
str.insert(4, 123);
System.out.println(str);
```

Output:

Test123

- **reverse()**

This method reverses the characters within a StringBuffer object.

Example:

```
StringBuffer str = new StringBuffer("Hello");
str.reverse();
System.out.println(str);
```

Output:

olleH

- **replace()**

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

Example:

```
StringBuffer str = new StringBuffer("Hello World");
str.replace( 6, 11, "GGI");
System.out.println(str);
```

Output:

Hello GGI

- **delete()**

The delete() method of StringBuffer class deletes the string from the specified beginIndex to endIndex.

Example:

```
StringBuffer strBuffer=new StringBuffer("HALDIA");
strBuffer.delete( 3, 6);
System.out.println(strBuffer);
```

Output:

HAL

- **capacity()**

The capacity() method returns the current capacity of StringBuffer object. The capacity is the amount of storage available for newly inserted characters, beyond which an allocation will occur.

Example:

```
StringBuffer strBuffer=new StringBuffer();

System.out.println(strBuffer.capacity());
strBuffer.append("1234");

System.out.println(strBuffer.capacity());
strBuffer.append("123456789112");

System.out.println(strBuffer.capacity());
strBuffer.append("1");
System.out.println(strBuffer.capacity()); //(oldcapacity*2)+2

StringBuffer strBuffer2=new StringBuffer("1234");
System.out.println(strBuffer2.capacity());
```

Output:

16 16 16 34 20

- **ensureCapacity()**

The ensureCapacity() method of StringBuffer class ensures that the given capacity is the minimum to the current capacity. If the current capacity is less than the argument, then a new internal array is allocated with greater capacity. The new capacity is the larger of:

- The minimumCapacity argument.
- Twice the old capacity, plus 2.

If the minimum Capacity argument is non-positive, this method takes no action and simply returns.

Example:

```
StringBuffer strBuffer=new StringBuffer("Core");
strBuffer.ensureCapacity(10);
System.out.println(strBuffer.capacity());

StringBuffer strBuffer2=new StringBuffer("Core");
strBuffer2.ensureCapacity(30);
System.out.println(strBuffer2.capacity());
```

Output:

```
20 42
```