Understanding String, StringBuilder and StringBuffer

String

We all know it. The String class in Java is the most basic way to represent and work with text. It is simple to use and provides several methods for string manipulation. However, it comes with a big limitation: Immutability. Once a String object is created, it cannot be modified. Any operation that appears to modidy a string actually creates a new string object.

Each time you perform an operation on a string (eg concatenation or substring extraction), a new string is created in memory. Puh. This can lead to increased memory usage, especially when working with large strings or when you perfrom multiple operations. But of course, there is help:

StringBuilder and StringBuffer

To address the limiations of the immutable String class, Java provides two other classes: StringBuilder and StringBuffer. These classes are designed for efficient string manipulation and dynamic content creation

StringBuilder

This is an unsynchronized, mutable sequence of characters. It is designed for single-threaded scenarios where performance is crucial. With **StringBuilder**, you can append, insert or delete characters within the same object, avoiding the need to create multiple intermediate string objects. Good for dynamic string construction and concatenation.

StringBuffer

Similar to StringBuilder, **StringBuffer** also provides mutable sequences of characters. However, unlike StringBuilder, **StringBuffer** is synchronized, making it thread-safe. This means that multiple threads can manipulate a StringBuffer object concurrently without causing data corruption. However, this thread safety comes at the cost of some performance.

It's worth noting that both StringBuilder and StringBuffer belong to the java.lang package. This means that you don't need to import any additional packages to use them; they are readily available for use in your Java programs.

Creating a string

- String is a group of characters.
- They are objects of type String class.
- Once a String object is created it cannot be changed i.e., Immutable.
- String are declared as final, so there cannot be subclasses of these classes.
- The default constructor creates an empty string.

```
String str = new String();
```

String s1 = **“welcome”;** //primitive type

equivalent to

String s2 = new String(“welcome”); //object type

or equivalent to

char data[] = {‘w’, ‘e’, ‘l’, ‘c’, ‘o’, ‘m’, ‘e’}; //array type

String s3 = new String(data);

or equivalent to

String s4 = new String(s1); //reference type

String Class Methods

1. The length() method returns the length of the string.

   Ex:

   - System.out.println(“Hari".length());
   - Output: 4.

2. The + operator is used to concatenate two or more strings.

   Ex:

   - String myName = “Kaven";
   - String s = "My name is" + myName+ ".";

- For string concatenation, the Java compiler converts an operand to a String whenever the other operand of the + is a String object.

3. charAt() method.

Characters in a string can be retrieved in a number of ways.

```
public char charAt(int index)
```

- Method returns the character at the specified index.
- An index ranges from 0 to length()-1

```
char c;
c = "abc".charAt(1);
// Output :
c = 'b'
```

4. **equals()** Method

- It compares the Strings. It returns true, if the argument is not null and it contain the same sequence of characters.

```
public boolean equals(anotherString);
String s1="VIT-AP";
String s2="vit-AP";
boolean result = s1.equals(s2);        //false is the result
```

5. **equalsIgnoreCase()** Method

- Compares two Strings, ignoring case considerations..

```
public boolean equalsIgnoreCase(anotherString);
String s1="VIT-AP";
String s2="vit-AP";
boolean result = s1.equalsIgnoreCase(s2);     //true is the result
```

6. startsWith()

- Checks the String starts with the specified prefix.

```
public boolean startsWith(String prefix)
"January".startsWith("Jan");
// Output: true
```

7. endsWith()

- Checks the String ends with the specified suffix.

```
public boolean endsWith(String suffix)
"January".endsWith("ry");
// Output: true
```

8. compareTo()

- Compares two strings and to know which string is bigger or smaller
- Returns negative integer, if this String object is less than the argument string
- Returns positive integer if this String object is greater than the argument string.
- return 0(zero), if these strings are equal.

```
public int compareTo(String anotherString)
```

9. public int compareToIgnoreCase(String str)

- This method is similar to compareTo() method, but this does not consider the case of strings.

10. indexOf() Method

- Searches for the first occurrence of a character or substring.
- Returns -1 if the character does not occur

public int indexOf(int ch)

- It searches for the character represented by ch within this string and returns the index of first occurrence of this character

public int indexOf(String str)

- It searches for the substring specified by str within this string and returns the index of first occurrence of this substring

Example:

```
String str = "How was your day today?";
str.indexOf('t');                    //17
str.indexOf("was");        //4
```

public int indexOf(int ch, int index)

- It searches for the character represented by ch within this String and returns the index of first occurrence of this **character** starting from the position specified by from index

public int indexOf(String str, int index)

- It searches for the substring represented by str within this String and returns the index of first occurrence of this **substring** starting from the position specified by from index

Example:

```
String str = "How was your day today?";
str.indexOf('a',6);          //14
str.indexOf("was",2);        //4
```

11.lastIndexOf()

- It searches for the last occurrence of a particular character or substring

12. substring()

- This method returns a new string which is actually a substring of this string.
- It extracts characters starting from the specified index all the way till the end of the string

public String substring(int beginIndex)

Eg:

“unhappy”.substring(2) returns “happy”

public String substring(int beginIndex, int endIndex)

Eg:

“smiles”.substring(1,5) returns “mile”

13.concat()

- Concatenates the specified string to the end of this string

public String concat(String str)

"to".concat("get").concat("her") //return together

14.replace()

- Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar

```
public String replace(char oldChar, char newChar);
String str = "How was your day today?";
System.out.println(str.replace('a', '3'));
    //displays How w3s your d3y tod3y?“
```

15.trim()

- Returns a copy of the string, with leading and trailing whitespace omitted

  public String trim()

  String s = “Hi Mom! “.trim();

  S = “Hi Mom!”

16. valueOf()

- This method is used to convert a character array into String.
- The result is a String representation of argument passed as character array

  public static String valueOf (char[] data);

valueOf()

This method is used to convert anything into String.

```
String s= String.valueOf(boolean b);
String s= String.valueOf(char c);
String s= String.valueOf(int i);
String s= String.valueOf(float f);
String s= String.valueOf(double d);
```

17.toLowerCase():

- Converts all characters in a String to lower case

```
public String toLowerCase();
String s = “JaVa”.toLowerCase();          // java
```

18.toUpperCase():

- Converts characters in a String to upper case

```
public String toUpperCase();
String s = “JaVa”.toLowerCase();          // JAVA
```

19. split() method can split a String into n number of sub strings.

```
public String[] split(String regex)
```

Ex:

```
public class StringEx1{
  public static void main(String [] args) {
   String s="Welcome to VIT-AP";
   String[] s1=s.split(" ");
   for(String i:s1)
       System.out.println(i);
}
}
```

```
D:\SWJava\bin>java StringEx1
Welcome
to
VIT-AP

D:\SWJava\bin>
```

20. toCharArray() method can split a String into n number of characters.

public char[] toCharArray()

Ex.

```
public class StringEx2{
public static void main(String [] args) {
 String s="Welcome";
 char[] c=s.toCharArray();
   for(char i:c)
   System.out.println(i);
}
}
```

```
D:\SWJava\bin>java StringEx2
W
e
l
c
o
m
e

D:\SWJava\bin>
```

21. contains() method used to check whether the given sub string is a part of the other string.

public boolean contains(String s)

Ex.

```
public class StringEx3{
public static void main(String [] args) {
 String s="VIT-AP";
 if(s.contains("VIT"))
     System.out.print("Welcome to VIT-AP");

}
}
```

```
D:\SWJava\bin>java StringEx3
Welcome to VIT-AP
D:\SWJava\bin>
```

22. format() method used to format any data into specified format of string.

public static String format(String format, Object... args)

Ex.

```
String str1 = String.format("%d", 101);              // Integer value
String str2 = String.format("%s", "Amar Singh");     // String value
String str3 = String.format("%f", 101.00);           // Float value
String str4 = String.format("%2.2f", 101.14325);     // Float value with 2 points
String str5 = String.format("%x", 10);               // Hexadecimal value
String str6 = String.format("%c", 'c');              // Char value
```

```
D:\SWJava\bin>java StringEx4
Integer value  101
Hexadecimal value  a
```

String Buffer Class

- String class is used to create a string of fixed length.
- But StringBuffer class creates strings of flexible length which can b modified.
- StringBuffer class objects are mutable, so they can be changeable
- StringBuffer are declared as final.
- StringBuffer class defines three constructors:
  1. StringBuffer()
     - Creates empty object with initial capacity of 16 characters.
  2. StringBuffer(int capacity)
     - Creates an **empty** (no character in it) object with a given capacity (not less than 0) for storing a string
  3. StringBuffer(String str)
     - Create StringBuffer object with string object.
     - The initial capacity of the string buffer is 16, plus the length of the string arguments.
     - The value of str cannot be null,

```
StringBuffer sb1=new StringBuffer();
StringBuffer sb2=new StringBuffer(5);
StringBuffer sb3=new StringBuffer("Welcome");

System.out.println(sb1.capacity());        // 16
System.out.println(sb1.length());          // 0
System.out.println(sb2.capacity());        // 10
System.out.println(sb2.length());          // 0
System.out.println(sb3.capacity());        // 23
System.out.println(sb3.length());          // 7
```

1. append() method - Append any data type with the string.

public StringBuffer append(boolean b)

Ex:

```
StringBuffer sb3=new StringBuffer("Welcome");
sb3.append(" VIT-AP");
System.out.println(sb3);                    // Welcome VIT-AP
```

2. insert() - insert any data into string at any location.

public StringBuffer insert(int offset, int i)

Ex:

```
StringBuffer sb3=new StringBuffer("Welcome VIT-AP");
sb3.insert(8, “to”);
System.out.println(sb3);
```

```
D:\SWJava\bin>java StringEx6
Welcome VIT-AP
Welcome toVIT-AP
```

3. delete() method - deletes a sub string from the specified begin index to end index

public StringBuffer delete(int start, int end)

Ex:

```
StringBuffer sb3=new StringBuffer("Welcome VIT-AP");
sb3.insert(8, “to”);
System.out.println(sb3);      // Welcome to VIT-AP
sb3.delete(8, 16);
System.out.println(sb3);              // Welcome
```

4. replace() –replaces part of this StringBuffer(substring) with another substring

public StringBuffer replace(int start, int end, String str);

Ex:

```
StringBuffer sb3=new StringBuffer("Welcome");
sb3.replace(3, 7, "done");
System.out.println(sb3);                    // Weldone
```

5. reverse() –reverses the StringBuffer and store it in the same StringBuffer object.

public StringBuffer replace(int start, int end, String str);

Ex:

```
StringBuffer sb3=new StringBuffer("Welcome");
sb3.reverse();
 System.out.println(sb3);    // emocleW
```

6. tostring() – Represents the object of StringBuffer class into String

```
StringBuffer sb3=new StringBuffer("Welcome");
sb3.reverse();
String s=sb3.toString();
System.out.println(s);
```

```
D:\SWJava\bin>java StringEx7
emocleW
```

Example1 : Forming Sentence Using StringBuilder Class

```
public class StrBuildDemo {
   public static void main(String[] args) {
      StringBuilder sentence = new StringBuilder();

      sentence.append("This");
      sentence.append(" is");
      sentence.append(" a");
      sentence.append(" StringBuilder");
      sentence.append(" example.");

      String s = sentence.toString();
      System.out.println(s);
   }
}
```

```
D:\AY2026_27\JAVA\LAB\Week7>javac StrBuildDemo.java

D:\AY2026_27\JAVA\LAB\Week7>java StrBuildDemo
This is a StringBuilder example.
```

Example 2: Reversing a String

```
import java.util.*;
public class Reverse{
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter the String : ");
      String original = sc.nextLine();
      StringBuilder r = new StringBuilder(original);

      r.reverse();
      String rs = r.toString();

      System.out.println("Original: " + original);
      System.out.println("Reversed: " + rs);
   }
}
```

```
D:\AY2026_27\JAVA\LAB\Week7>javac Reverse.java

D:\AY2026_27\JAVA\LAB\Week7>java Reverse
Enter the String :
LBRCE
Original: LBRCE
Reversed: ECRBL
```

Example 3: Anagram Check — String

```
import java.util.*;
public class Anagram {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter First String : ");
      String str1 = sc.nextLine();
      System.out.print("Enter Second String : ");
      String str2 = sc.nextLine();

      boolean status = areAnagrams(str1, str2);
      if(status)
         System.out.println("YES, The strings are anagrams ");
      else
         System.out.println("No, The strings are NOT anagrams ");
   }

   public static boolean areAnagrams(String str1, String str2) {
      if (str1.length() != str2.length()) {
         return false;
      }

      int[] charCount = new int[256]; // Assuming ASCII characters

      for (char c : str1.toCharArray()) {
         charCount[c]++;
      }

      for (char c : str2.toCharArray()) {
         charCount[c]--;
      }

      for (int count : charCount) {
         if (count != 0) {
```

```
            return false;
        }
    }

    return true;
  }
}
```

```
D:\AY2026_27\JAVA\LAB\Week7>javac Anagram.java

D:\AY2026_27\JAVA\LAB\Week7>java Anagram
Enter First String : silent
Enter Second String : listen
YES, The strings are anagrams

D:\AY2026_27\JAVA\LAB\Week7>java Anagram
Enter First String : LBRCE
Enter Second String : lbrce
No, The strings are NOT anagrams
```

Example 4: String Palindrome

```
import java.util.Scanner;
public class StringPalindrome {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter a String: ");
      String str = sc.nextLine();
      String reverse = "";
      for (int i = str.length() - 1; i >= 0; i--) {
         reverse = reverse + str.charAt(i);
      }
      if (str.equals(reverse)) {
         System.out.println("The String is a Palindrome");
      } else {
         System.out.println("The String is NOT a Palindrome");
      }
      sc.close();
   }
}
```

```
D:\AY2026_27\JAVA\LAB\Week7>java StringPalindrome
Enter a String: level
The String is a Palindrome

D:\AY2026_27\JAVA\LAB\Week7>java StringPalindrome
Enter a String: hello
The String is NOT a Palindrome
```

Example 5: Even length words in a String

```
import java.util.Scanner;
public class EvenLengthWords {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter a String: ");
      String str = sc.nextLine();
      String[] words = str.split(" ");
      System.out.println("Even length words are:");
      for (String word : words) {
         if (word.length() % 2 == 0) {
            System.out.println(word);
         }
      }
      sc.close();
   }
}
```

```
D:\AY2026_27\JAVA\LAB\Week7>java EvenLengthWords
Enter a String: this is an example for even length words
Even length words are:
this
is
an
even
length
```

Example 6: Count the number of words in a String

```
import java.util.Scanner;

public class WordCount {
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a String: ");
        String str = sc.nextLine();
        String[] words = str.trim().split("\\s+");
        int count = words.length;
        System.out.println("Number of words: " + count);
        sc.close();
    }
}
```

```
D:\AY2026_27\JAVA\LAB\Week7>java WordCount
Enter a String: this is an example for word count
Number of words: 7
```

Example 7: Find the maximum and minimum occurring character in a String

```
import java.util.Scanner;
public class MaxMinCharacter {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a String: ");
        String str = sc.nextLine().toLowerCase();
        int[] count = new int[256];

        for (int i = 0; i < str.length(); i++) {
            char ch = str.charAt(i);
            if (ch != ' ') {
                count[ch]++;
            }
        }

        int max = 0;
        int min = Integer.MAX_VALUE;
        for (int i = 0; i < 256; i++) {
            if (count[i] > 0) {

                if (count[i] > max) {
                    max = count[i];
                }
```

```
            if (count[i] < min) {
                min = count[i];
            }
        }
    }

    System.out.println("\nMaximum occurring character(s):");
    for (int i = 0; i < 256; i++) {
        if (count[i] == max) {
            System.out.println((char)i + " -> " + count[i]);
        }
    }
    System.out.println("\nMinimum occurring character(s):");
    for (int i = 0; i < 256; i++) {
        if (count[i] == min) {
            System.out.println((char)i + " -> " + count[i]);
        }
    }
    sc.close();
  }
}
```

```
D:\AY2026_27\JAVA\LAB\Week7>java MaxMinCharacter
Enter a String: This is an Example for Maximum and Minimum occuring Characters in a String

Maximum occurring character(s):
i -> 8

Minimum occurring character(s):
d -> 1
f -> 1
l -> 1
p -> 1
```

Example 8: Remove all the duplicates from given string

```
import java.util.Scanner;
public class RemoveDuplicates {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter a String: ");
    String str = sc.nextLine();
    String result = "";
    for (int i = 0; i < str.length(); i++) {
      char ch = str.charAt(i);
```

```
            if (result.indexOf(ch) == -1) {
                result = result + ch;
            }
        }
        System.out.println("String after removing duplicates: " + result);
        sc.close();
    }
}
```

```
D:\AY2026_27\JAVA\LAB\Week7>java RemoveDuplicates
Enter a String: missisipi
String after removing duplicates: misp
```

Example 9: Sort The Strings

```
import java.util.Scanner;
import java.util.Arrays;
public class SortCities {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of cities: ");
        int n = sc.nextInt();
        sc.nextLine(); // consume newline
        String[] cities = new String[n];
        for (int i = 0; i < n; i++) {
            System.out.print("Enter city " + (i + 1) + ": ");
            cities[i] = sc.nextLine();
        }
        Arrays.sort(cities);
        System.out.println("\nCities in alphabetical order:");
        for (String city : cities) {
            System.out.println(city);
        }
        sc.close();
    }
}
```

```
D:\AY2026_27\JAVA\LAB\Week7>javac SortCities.java

D:\AY2026_27\JAVA\LAB\Week7>java SortCities
Enter the number of cities: 5
Enter city 1: Vijayawada
Enter city 2: Guntur
Enter city 3: Hyderabad
Enter city 4: Madras
Enter city 5: Chennai

Cities in alphabetical order:
Chennai
Guntur
Hyderabad
Madras
Vijayawada
```

Example 10: Extract comma separated word from String

```
import java.util.StringTokenizer;
public class StringTokenizerDemo {
   public static void main(String[] args) {

      String str = "Java,C,Python,C++";

      StringTokenizer st = new StringTokenizer(str, ",");

      while (st.hasMoreTokens()) {
         System.out.println(st.nextToken());
      }
   }
}
```

```
D:\AY2026_27\JAVA\LAB\Week7>javac StringTokenizerDemo.java

D:\AY2026_27\JAVA\LAB\Week7>java StringTokenizerDemo
Java
C
Python
C++
```

Feature	String	StringBuffer	StringBuilder
Mutability	The `String` class creates immutable objects, meaning once a string is created, it cannot be modified. Any modification results in the creation of a new object.	The `StringBuffer` class creates mutable objects, meaning the content can be modified without creating a new object.	The `StringBuilder` class also creates mutable objects, allowing changes to be made directly without creating new objects.
Thread-Safety	The `String` class is inherently thread-safe because it is immutable, so multiple threads can share it without any issues.	The `StringBuffer` class is synchronized, which means it is thread-safe and can be used safely in multi-threaded environments.	The `StringBuilder` class is not synchronized, meaning it is not thread-safe and should only be used in single-threaded environments.
Performance	The `String` class is slower for repeated modifications because each modification creates a new object in memory.	The `StringBuffer` class is slower than `StringBuilder` because synchronization adds extra overhead.	The `StringBuilder` class is faster than `StringBuffer` because it does not have synchronization overhead.
Package	The `String` class is present in the `java.lang` package.	The `StringBuffer` class is present in the `java.lang` package.	The `StringBuilder` class is present in the `java.lang` package.
Introduced In	The `String` class was introduced in **JDK 1.0**.	The `StringBuffer` class was introduced in **JDK 1.0**.	The `StringBuilder` class was introduced in **JDK 1.5**.
Usage	The `String` class is used when the string content will remain constant and no modifications are required.	The `StringBuffer` class is used when strings need to be modified frequently in a multi-threaded environment.	The `StringBuilder` class is used when strings need to be modified frequently in a single-threaded environment for better performance.
Example	`String s = "Hello";`	`StringBuffer sb = new StringBuffer("Hello");`	`StringBuilder sb = new StringBuilder("Hello");`