String vs StringBuilder vs StringBuffer in Java

**Prerequisites :**[String](https://www.geeksforgeeks.org/string-class-in-java/) , [Initialize a String](https://www.geeksforgeeks.org/how-to-initialize-and-compare-strings-in-java/)

Consider below code with three concatenation functions with three different types of parameters, String, StringBuffer and StringBuilder.

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| // Java program to demonstrate difference between String,  // StringBuilder and StringBuffer  class Geeksforgeeks  {      // Concatenates to String      public static void concat1(String s1)      {          s1 = s1 + "forgeeks";      }        // Concatenates to StringBuilder      public static void concat2(StringBuilder s2)      {          s2.append("forgeeks");      }        // Concatenates to StringBuffer      public static void concat3(StringBuffer s3)      {          s3.append("forgeeks");      }        public static void main(String[] args)      {          String s1 = "Geeks";          concat1(s1);  // s1 is not changed          System.out.println("String: " + s1);            StringBuilder s2 = new StringBuilder("Geeks");          concat2(s2); // s2 is changed          System.out.println("StringBuilder: " + s2);            StringBuffer s3 = new StringBuffer("Geeks");          concat3(s3); // s3 is changed          System.out.println("StringBuffer: " + s3);      }  } |

**Output:**

String: Geeks

StringBuilder: Geeksforgeeks

StringBuffer: Geeksforgeeks

**Explanation:**  
**1. Concat1**: In this method, we pass a string “Geeks” and perform “s1 = s1 + ”forgeeks”. The string passed from main() is not changed, this is due to the fact that String is **immutable**. Altering the value of string creates another object and s1 in concat1() stores reference of new string. References s1 in main() and cocat1() refer to different strings.

**2. Concat2**: In this method, we pass a string “Geeks” and perform “s2.append(“forgeeks”)” which changes the actual value of the string (in main) to “Geeksforgeeks”. This is due to the simple fact that StringBuilder is **mutable** and hence changes its value.

**2. Concat3**: StringBuffer is similar to StringBuilder except one difference that StringBuffer is thread safe, i.e., multiple threads can use it without any issue. The thread safety brings a penalty of performance.

**When to use which one :**

* If a string is going to remain constant throughout the program, then use String class object because a String object is immutable.
* If a string can change (example: lots of logic and operations in the construction of the string) and will only be accessed from a single thread, using a StringBuilder is good enough.
* If a string can change, and will be accessed from multiple threads, use a StringBuffer because StringBuffer is synchronous so you have thread-safety.