

[JAVA TUTORIAL](#)[#INDEX POSTS](#)[#INTERVIEW QUESTIONS](#)[RESOURCES](#)[HIRE ME](#)[DOWNLOAD ANDROID APP](#)[CONTRIBUTE](#)**Subscribe to Download Java Design Patterns eBook****DOWNLOAD NOW**[HOME](#) » [INTERVIEW QUESTIONS](#) » [STRING VS STRINGBUFFER VS STRINGBUILDER](#)

String vs StringBuffer vs StringBuilder

APRIL 2, 2018 BY [PANKAJ](#) — [5 COMMENTS](#)

Today we will look into difference between String vs [StringBuffer vs StringBuilder](#). StringBuffer vs StringBuilder is a popular [java interview](#) question.

Table of Contents [\[hide\]](#)

[1 String vs StringBuffer vs StringBuilder](#)

[1.1 String in Java](#)

[1.2 String vs StringBuffer](#)

[1.3 StringBuffer vs StringBuilder](#)

[1.4 String vs StringBuffer vs StringBuilder](#)

String vs StringBuffer vs StringBuilder

Difference between String, StringBuffer and StringBuilder

String is one of the most widely used class in java programming and one of the most important topic in core java interview. If you are writing a program which prints something on console, we use String.

This tutorial is aimed to focus on major features of String class and then we will compare **String vs StringBuffer vs StringBuilder**.

String in Java

1. String class represents character strings, we can instantiate String by two ways.
`String str = "abc";` or `String str = new String ("abc");`
2. String is **immutable** in java, so its easy to share it across different threads or functions.
3. When we create a String using double quotes, it first looks for the String with same value in the JVM string pool, if found it returns the reference else it creates the String object and then place it in the String pool. This way JVM saves a lot of space by using same String in different threads. But if new operator is used, it explicitly creates a new String in the heap memory.
4. + operator is overloaded for String and used to concatenate two Strings. Although internally it uses StringBuffer to perform this action.
5. String overrides equals() and hashCode() methods, two Strings are equal only if they have same characters in same order. Note that equals() method is case sensitive, so if you are not looking for case sensitive checks, you should use equalsIgnoreCase() method.
6. A String represents a string in the UTF-16 format
7. String is a final class with all the fields as final except "private int hash". This field contains the hashCode() function value and created only when hashCode() method is called and then cached in this field. Furthermore, hash is generated using final fields of String class with some calculations, so every time hashCode() method is called, it will result in same output. For caller, its like calculations are happening every time but internally it's cached in hash field.

String vs StringBuffer

Since String is immutable in java, whenever we do String manipulation like concat, substring etc, it generates a new String and discard the older String for garbage collection.

These are heavy operations and generate a lot of garbage in heap. So Java has provided StringBuffer and StringBuilder class that should be used for String manipulation.

StringBuffer and StringBuilder are mutable objects in java and provide append(), insert(), delete() and substring() methods for String manipulation.

StringBuffer vs StringBuilder

StringBuffer was the only choice for String manipulation till Java 1.4 but it has one disadvantage that all of its public methods are synchronized. StringBuffer provides Thread safety but on a performance cost.

In most of the scenarios, we don't use String in multithreaded environment, so Java 1.5 introduced a new class StringBuilder that is similar with StringBuffer except thread safety and synchronization.

So if you are in a single threaded environment or don't care about thread safety, you should use StringBuilder else use StringBuffer. See this post for [performance benchmarking between StringBuffer and StringBuilder](#).

String vs StringBuffer vs StringBuilder

1. String is immutable whereas StringBuffer and StringBuilder are mutable classes.
2. StringBuffer is thread safe and synchronized whereas StringBuilder is not, that's why [StringBuilder is more faster than StringBuffer](#).
3. String concat + operator internally uses StringBuffer or StringBuilder class.
4. For String manipulations in non-multi threaded environment, we should use StringBuilder else use StringBuffer class.

That's all for a quick roundup of difference between String vs StringBuffer vs StringBuilder. In most of the scenarios for string manipulation, StringBuilder is better suited than StringBuffer.

References:

- [String API Doc](#)
- [StringBuffer API Doc](#)
- [StringBuilder API Doc](#)
- [StringBuffer vs StringBuilder](#)

**« PREVIOUS**

Longest Palindrome in a String in Java

NEXT »

String Concatenation in Java

About Pankaj

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on [Google Plus](#), [Facebook](#) or [Twitter](#). I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on [Youtube](#).

FILED UNDER: [INTERVIEW QUESTIONS](#), [JAVA](#)

Comments

emilio says[AUGUST 20, 2014 AT 11:16 AM](#)

+ operator is overloaded for String and used to concatenate two Strings. Although internally it uses StringBuffer to perform this action.

As I know internally it is uses StringBuilder to perform this action.

[Example here](#)

[Reply](#)

subrat panda says

JULY 23, 2013 AT 2:47 PM

If we look both String and StringBuilder both are final class and fields are also final, then what extra property is there in String class, that make it immutable and StringBuilder as immutable. Please provide me the solution.

Q2:-When we are creating our custom immutable class, generally we are making our class as

1)final,

2)all the fields are making final and initializing them with the help of constructor

3) only providing the getter methods, no setter methods

It will holds good for primitive type data Type or immutable class(StringBuffer), what if in our class there is a mutable property is there, how to achieve immutable in that case.

Please provide me the answer with example.

From a long time I was looking for this answer.

nb:- from interet I found something like make the defense copy of that property, but I didn't get how exactly I ll achieve that.

[Reply](#)

Pankaj says

JULY 24, 2013 AT 3:30 AM

String is immutable because once we instantiate it, there is no way to change it's value whereas StringBuffer and StringBuilder is mutable because they provide append() method to update its content.

If your custom immutable class has a mutable field, you should return the defensive copy of that variable in get() method. For example, let's say you have a List<String> variable in your class, so your get method should be something like

```
public List<String> get(){
    return (List<String>) myList.clone();
}
```

So now even if you update the contents of the returned List, it won't update the content of list in our immutable object.

Read this post for further details: <https://www.journaldev.com/129/how-to-write-an-immutable-class>

[Reply](#)

Mikhail Seleznev says

JANUARY 26, 2013 AT 4:28 PM

"But if new operator is used, it explicitly creates a new String and then add it to the pool."
Operator new creates a new String but doesn't add String to the pool. Method intern() does that.

[Reply](#)**Pankaj says**

JANUARY 29, 2013 AT 1:57 AM

Good observation, updated the post.

[Reply](#)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

☐

Save my name, email, and website in this browser for the next time I comment.

POST COMMENT

DOWNLOAD ANDROID APP



CORE JAVA TUTORIAL

Java 10 Tutorials Java 9 Tutorials
Java 8 Tutorials Java 7 Tutorials Core
Java Basics OOPS Concepts Data
Types and Operators String Manipulation
Java Arrays Annotation and Enum
Java Collections Java IO Operations
Java Exception Handling
MultiThreading and Concurrency
Regular Expressions Advanced Java
Concepts

IMPORTANT INTERVIEW QUESTIONS

Java Interview Questions

- > [Core Java Interview Questions](#)
- > [String Interview Questions](#)
- > [Multithreading Interview Questions](#)
- > [Collections Interview Questions](#)
- > [Exception Interview Questions](#)
- > [Java Programming Interview Questions](#)

- > [Java 8 Interview Questions Part 1](#)
- > [Java 8 Interview Questions Part 2](#)
- > [Servlet Interview Questions](#)
- > [JSP Interview Questions](#)
- > [Struts 2 Interview Questions](#)
- > [JDBC Interview Questions](#)
- > [Spring Interview Questions](#)
- > [Hibernate Interview Questions](#)
- > [JSF Interview Questions](#)
- > [Web Services Interview Questions](#)
- > [Scala Basic Interview Questions](#)
- > [Scala Intermediate Interview Questions](#)
- > [Scala Advanced Interview Questions](#)
- > [Scala Interview Questions Summary](#)
- > [Common Job Interview Questions](#)

Miscellaneous

- > [Java ClassLoader](#)
- > [String StringBuffer StringBuilder](#)
- > [Java is Pass By Value](#)
- > [Java Heap vs Stack Memory](#)

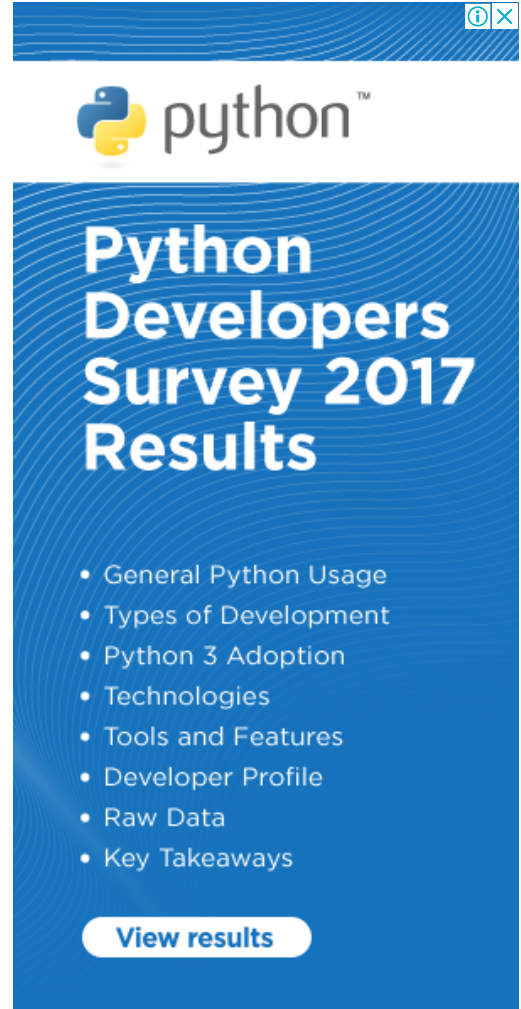
RECOMMENDED TUTORIALS

Java Tutorials

- > [Java IO](#)
- > [Java Regular Expressions](#)
- > [Multithreading in Java](#)
- > [Java Logging](#)
- > [Java Annotations](#)
- > [Java XML](#)
- > [Collections in Java](#)
- > [Java Generics](#)
- > [Exception Handling in Java](#)
- > [Java Reflection](#)
- > [Java Design Patterns](#)
- > [JDBC Tutorial](#)

Java EE Tutorials

- > [Servlet JSP Tutorial](#)
- > [Struts2 Tutorial](#)
- > [Spring Tutorial](#)
- > [Hibernate Tutorial](#)
- > [Primefaces Tutorial](#)
- > [Apache Axis 2](#)
- > [JAX-RS](#)
- > [Memcached Tutorial](#)



The banner features the Python logo at the top left. The main title 'Python Developers Survey 2017 Results' is prominently displayed in white text against a blue background with a subtle wave pattern. Below the title, a list of survey topics is provided, and a 'View results' button is located at the bottom right.

python™

Python Developers Survey 2017 Results

- General Python Usage
- Types of Development
- Python 3 Adoption
- Technologies
- Tools and Features
- Developer Profile
- Raw Data
- Key Takeaways

[View results](#)