**-------------------------------------------------------------------------------------------------------------**

**Core Java**

**-------------------------------------------------------------------------------------------------------------**

**Java 8 Features 🡪**

1.Lambda expression

2.Stream API

3.Collections

**1.Lambda expression** 🡪 A lambda expression is a short block of code which takes in parameters and returns a value. Lambda expressions are similar to methods, but they do not need a name and they can be implemented right in the body of a method (i.e. no need to function call).

**2.Stream API 🡪**

🡪 to process huge amount of data.

stream()🡪stream methods comes under Collection.java class. Stream method returns a stream of value.

🡪The purpose of creating stream of data is we are not directly manipulating the original data we create stream of data and manipulate that stream , so in future if the data processing goes wrong then we have the original data for backup.

* Once we consume the stream we can not reuse it.

parallelStream() 🡪 to create multiple threads to process huge amount of data.

Ex 🡪

List<Integer> nums = Arrays.asList(4,5,6,7,8);

Stream<Integer> data = nums.stream();

data.forEach(n -> System.out.println(n));

**Wrapper Class 🡪** Wrapper classes are used to wrap or convert primitive data type into Objects.

🡪Data structures in the Collection framework, such as [ArrayList](https://www.geeksforgeeks.org/arraylist-in-java/) and [Vector](https://www.geeksforgeeks.org/vector-vs-arraylist-java/), store only objects (reference types) and not primitive types. So to store primitive data as object we need wrapper classes.



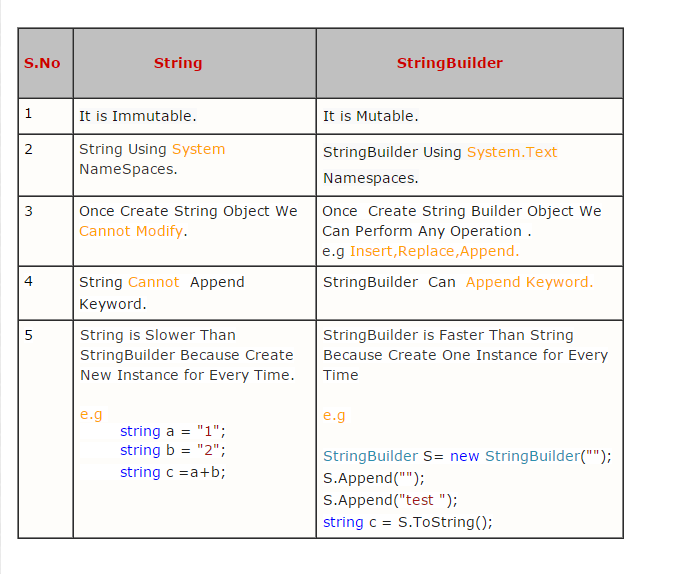


The main difference between the [.equals() method](https://www.geeksforgeeks.org/method-class-equals-method-in-java/) and ==[operator](https://www.geeksforgeeks.org/operators-in-java/)

* We can use == operators for reference comparison (**address comparison**) and .equals() method for **content comparison**. In simple words, == checks if both objects point to the same memory location whereas .equals() evaluates to the comparison of values in the objects.

<https://www.geeksforgeeks.org/difference-between-and-equals-method-in-java/>

|  |  |  |
| --- | --- | --- |
| **No.** | **String** | **StringBuffer** |
| 1) | The String class is immutable. | The StringBuffer class is mutable. |
| 2) | String is slow and consumes more memory when we concatenate too many strings because every time it creates new instance. | StringBuffer is fast and consumes less memory when we concatenate t strings. |
| 3) | String class overrides the equals() method of Object class. So you can compare the contents of two strings by equals() method. | StringBuffer class doesn't override the equals() method of Object class. |
| 4) | String class is slower while performing concatenation operation. | StringBuffer class is faster while performing concatenation operation. |
| 5) | String class uses String constant pool. | StringBuffer uses Heap memory |



|  |  |  |
| --- | --- | --- |
| **No.** | **StringBuffer** | **StringBuilder** |
| 1) | StringBuffer is *synchronized* i.e. thread safe. It means two threads can't call the methods of StringBuffer simultaneously. | StringBuilder is *non-synchronized* i.e. not thread safe. It means two threads can call the methods of StringBuilder simultaneously. |
| 2) | StringBuffer is *less efficient* than StringBuilder. | StringBuilder is *more efficient* than StringBuffer. |
| 3) | StringBuffer was introduced in Java 1.0 | StringBuilder was introduced in Java 1.5 |
| 4) | Exist in java.util.\* package | Exist in java.lang.\* package |

**StringBuilder 🡪** A String can be used when immutability is required, or concatenation operation is not required. A StringBuilder can be used when a mutable string is needed without the performance overhead of constructing lots of strings along the way.

**String Functions 🡪**

**Str = “hello , aakanksha”;**

**1. length() 🡪 str.length() 🡪 gives string length**

**2. equals() 🡪 str1.equals(str2) 🡪 gives true or false**

**3. equalsIngnoreCase() 🡪 str1.equalsIngnoreCase(str2) 🡪 compare two strings ignoring their cases.**

**4. charAt() 🡪 charAt(i) 🡪 Returns the character at specified index**

**5. replace() 🡪 str.replace(‘a’ , ’b’)**

**6. split() 🡪 str.split(‘ , ’) 🡪 split a string into an array of substrings**

**7. concat() 🡪 s.concat(‘abc’)**

**8. indexOf() 🡪 str.indexOf(‘a’) 🡪 returns the position of the first found occurrence of specified character in string.**

**9. contains() 🡪 str.contains(“hel”) 🡪 returns Boolean value**

**10. endsWith() 🡪 str.endsWith(“hel”)🡪 returns Boolean value**

**11. toCharArray()**

**12. toLowerCase()**

**13.toUpperCase()**

**StringBuilder functions**

1. **reverse()**
2. **append()**

# Why pointer concept not use in java?

By not allowing pointers, Java effectively provides another level of **abstraction** to the developer. No pointer support makes Java more secure because they point to memory location or used for **memory management** that loses the security as we use them directly.

[**https://net-informations.com/java/cjava/pointers.htm#:~:text=No%20pointer%20support%20make%20Java,variable%20in%20the%20caller's%20scope**](https://net-informations.com/java/cjava/pointers.htm#:~:text=No%20pointer%20support%20make%20Java,variable%20in%20the%20caller's%20scope)**.**

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Spring Framework** **🡪** it is an open source application framework that provides infrastructure support for developing java application. Spring helps developers create high performing applications using plain old Java objects (POJOs).

* Used for loose coupling of data

**Spring Core 🡪**

**12)** **What is the difference between Spring MVC and Spring core?**  
The Spring MVC is part of the Spring framework, which helps you develop Java web applications using model web controller patterns. At the same time, Spring Core provides the Dependency injection and Inversion of Control. The Spring Container is part of Spring core.  
  
Both functionalities come in different JAR files. If you are developing just a core Java application using Spring, you just need Spring Core, but if you are creating a Web application, then you need spring-mvc.jar as well.  
  
Read more: <https://www.java67.com/2012/08/spring-interview-questions-answers.html#ixzz7yAySZSCP>

**Spring MVC flow 🡪** The core component of Spring MVC is the DispatcherServlet class which handles user requests and then forwards them to the correct controller. This allows the controller to process the request, create the model and then provide the information to the end-user via a specified view.

1.**What does REST stand for?**

REST stands for the Representational State Transfer, which uses the HTTP protocol to send data from the client to the server. And the server can response to the client using JSON or XML.

HTTP also defines the following standard status code:

* **404:** RESOURCE NOT FOUND
* **200:** SUCCESS
* **201:** CREATED
* **401:** UNAUTHORIZED
* **500:** SERVER ERROR