1. Can somebody clarify what is meant by *immutable*?
2. Why is a String immutable?
3. What are the advantages/disadvantages of the immutable objects?
4. Why should a mutable object such as StringBuilder be preferred over String and vice-verse?
5. Why String class is final class?

An immutable object is an object that will not change its internal state after creation.

*Immutable objects* are very useful in multithreaded applications because they *can be shared between threads without synchronization*.

Immutable objects are always thread safe.

**To create an immutable object what rules we need to follow?**

To create an immutable object you need to follow some simple rules:

* Don't add any setter method
* Declare all fields final and private
* If a field is a mutable object create defensive copies of it for getter methods
* If a mutable object passed to the constructor must be assigned to a field create a defensive copy of it
* Don't allow subclasses to override methods.

Source- <https://dzone.com/articles/immutable-objects-in-java>

Mainly objects are 2 types-

**mutable &**

**immutable**

*Mutable objects are those objects whose content can be modified.*

*& Immutable objects are those objects once created cannot be modified.*

*& String class objects are immutable.*

Check below program – ImmutableStringEx.java

Github link- <https://github.com/jainabhishek0406/JavaString_ImmutabilityEx>

*/\*\*  
 \** ***@author*** *Abhishek Jain  
 \*  
 \* String is Immutable or Mutable???  
 \*/*public class ImmutableStringEx {  
 public static void main(String[] args) {  
 String str1 = "Abhishek";  
 String str2 = "Jain";  
  
 System.*out*.println("str1 & str2 values before string concatenation");  
 System.*out*.println("str1 = " + str1);  
 System.*out*.println("str2 = " + str2);  
  
 //string concatenation  
 str1 = str1+str2;  
  
 System.*out*.println("\nstr1 & str2 values before string concatenation");  
 System.*out*.println("str1 = " + str1);  
 System.*out*.println("str2 = " + str2);  
  
 }  
}

Output of this program is

str1 & str2 values before string concatenation

str1 = Abhishek

str2 = Jain

str1 & str2 values before string concatenation

str1 = AbhishekJain

str2 = Jain

It seems object str1 content is modified from ‘Abhishek’ to ‘AbhishekJain’, so str1 appears to be Mutable.

But what we are thinking here is incorrect, I already mentioned string is immutable, let me explain again.

It is called ‘unreferenced object’, and garbage collector will remove it from memory.

Str1

Abhishek

After concatenation, JVM creates a new object and store the string ‘AbhishekJain’ in that object. Basically it does not modify the content of str1. Only reference changed. And old object that contains ‘Abhishek’ has lost its reference.

AbhishekJain

Jain

Str2

Java did not provide the methods in String class which modify the contents

of the same objects.

And whenever we try to modify the contents, JVM will create a new object with

modified data. It will not modify the same object. Also there should not be any

scope to override those String class methods, this is the reason,

String class is made ‘final’ class. The methods of a final class cannot be overridden.