OBJECT™ TECHNOLOGIES

Strings in Java

www.object.co.in

What will be covered

- class String
- String immutability
- String pooling
- StringBuilder and StringBuffer as alternatives
- Imp methods in String class
- Common programming scenarios

class String

OBJECT

- String is a sequence of characters, for e.g. "Hello" is a string of 5 characters. In java, string is an immutable object which means it is constant and can cannot be changed once it has been created.
- Strings can be created by assigning a String literal to a String instance:

```
String str1 = "Welcome";
```

- String can be created using new keyword

```
String str1 = new String("Welcome");
```

- String manipulation is one of the most common activities in computer programming. String class has a variety of methods for string manipulation.
- These functionalities include getting the character at perticular index, concatenating, replacing the character, trimming etc

String immutabilty

OBJECT TECHNOLOGIES

- String class in Java is immutable. The meaning of immutable is unchangeable or unmodifiable.
- That is, once we create a string object with value, we are not allowed to perform any changes in that object.
- In other words, we cannot modify the value of the string. But if you try to change with a new value, a new string object will be created by storing a new value.
- So, we cannot perform any changes with the existing string object. This non-changeable behavior is nothing but an immutability concept in Java.
- Java implements this immutability concept to minimize the duplication of string values that tend to exist many times in any application program.

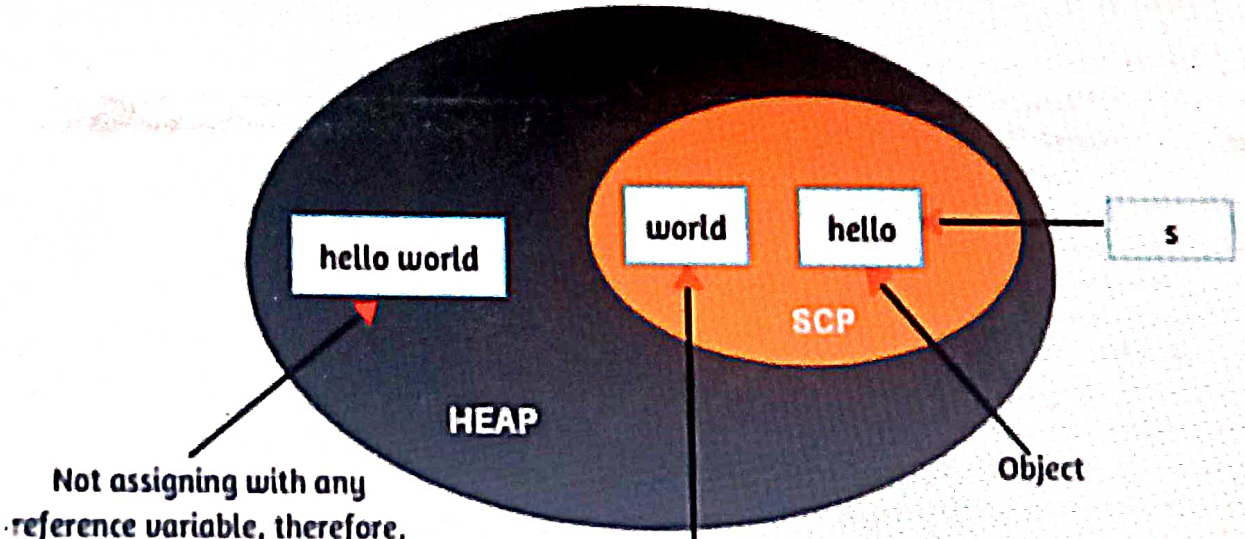
```
String s = "hello";

s.concat("world"); // concat() method adds string at the end.

System.out.println(s); // It will print "hello" because string is
immutable object.
```

String immutabilty

OBJECT TECHNOLOGIES



As you can see that three objects are created but s reference variable still refers to "hello", not to "hello world".

Fig: Allotting memory for storing object.

- Thus, the value of string s is not modified and still, 's' is pointing to "hello" only. Therefore, the result is "hello". This is the reason, string objects are called immutable in Java.

String immutabilty

- String objects are immutable in Java because Java uses the concept of string constant pool. Suppose there are 6 reference variables, and pointing to the same object "Hello world".
- If one reference variable of them changes the value of object from "Hello world" to "Hello", with this change, all the reference variables will be affected. That's why string objects are immutable in Java.
- The primary advantage of string immutability is that Java compiler can save space in the memory by sharing strings.

String pooling

OBJECT TECHNOLOGIES

- The String is the most widely used data structure. Caching the String literals and reusing them saves a lot of heap space because different String variables refer to the same object in the String pool. String intern pool serves exactly this purpose.
- Java String Pool is the special memory region where Strings are stored by the JVM. Since Strings are immutable in Java, the JVM optimizes the amount of memory allocated for them by storing only one copy of each literal String in the pool. This process is called interning:

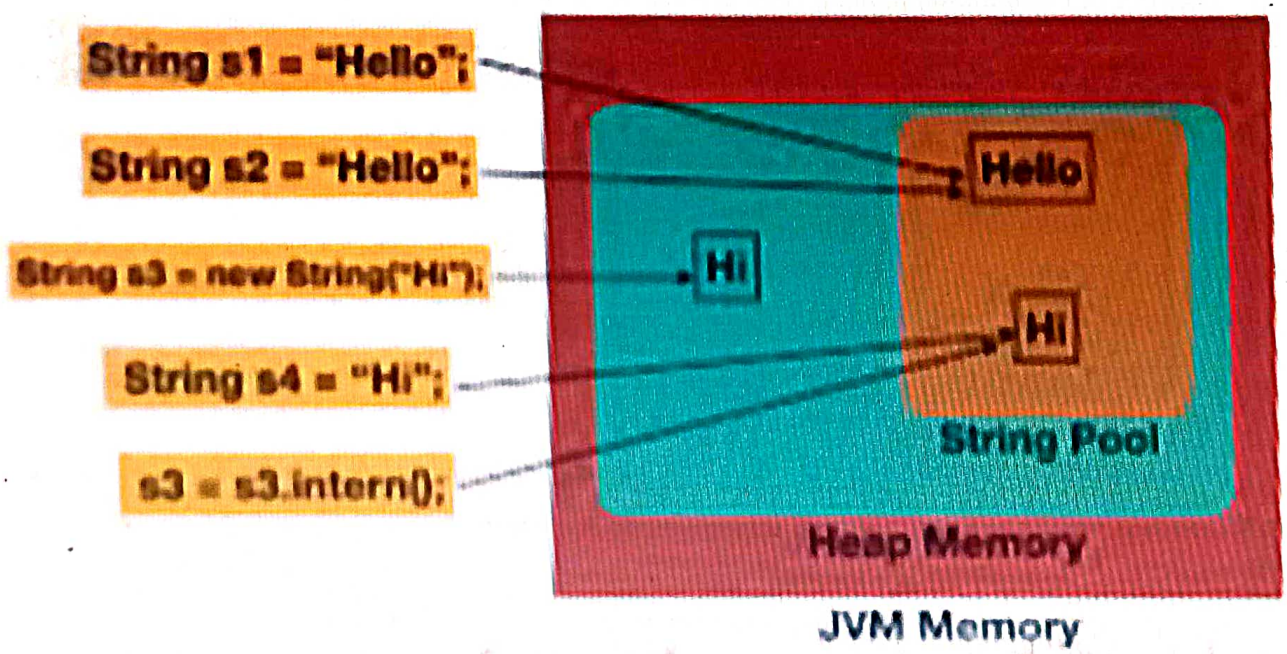
```
String s1 = "Hello World";

String s2 = "Hello World";

System.out.println(s1 == s2);    //prints true
```

- Because of the presence of the String pool in the preceding example, two different variables are pointing to same String object from the pool, thus saving crucial memory resource.

String pooling



StringBuilder and StringBuffer as alternatives

OBJECT TECHNOLOGIES

- Since String is immutable in Java, whenever we do String manipulation like concatenation, substring, etc. it generates a new String and discards 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 classes that should be used for String manipulation. StringBuffer and StringBuilder are mutable objects in Java. They provide append(), insert(), delete(), and substring() methods for String manipulation.
- StringBuffer provides Thread safety but at a performance cost. In most of the scenarios, we don't use String in a multithreaded environment.
- So Java 1.5 introduced a new class StringBuilder, which is similar to StringBuffer except for thread-safety and synchronization.
- If you are in a single-threaded environment or don't care about thread safety, you should use StringBuilder. Otherwise, use StringBuffer for thread-safe operations.

Imp methods in String class

OBJECT TECHNOLOGIES

String method	Purpose
public char charAt(int index)	To get chracter at specified index
public String concat(String s)	Appening s at the end of source string
public boolean equalsIgnoreCase(String s)	Source string string given sstring compared without consideration of cases
public int compareTo(String s)	compares the two strings based on the Unicode value of each character
public indexOf(char ch)	Returns the index of first occurrence of the specified character
public String substring(int index)	returns the substring of the string.
public int length()	returns the length of the String
public String replace(char old, char new)	any occurrence of the char in the first argument is replaced by the char in the second argument
public boolean contains("searchString")	method returns true of target String is containing search String