String

## String:

String is nothing but a sequence of characters. In java, strings are considered as Objects.

The string class is **immutable.** i.e. Once we created a string object we are not allowed to perform any operations on the existing string object. If we are trying to perform any changes with those changes a new string object will be created.

## Key Points:

String internally uses flyweight design pattern.

In String class, equals method overridden for content comparison.

We can also perform string concatenation by using ‘+’ operator apart from concat() method.

It is not a good programming practice to store the passwords in the String objects.

String objects are thread safe.

Hashcode() and equals() methods are overridden.

toString() method overridden to return the contents of the string.

It is introduced in 1.0version.

## Ways to create string object

### String literal

Example: String s1 = “javaByDeveloper”;

Here object will be created in string constant pool. Here s1 refers to SCP.

### Using new keyword

Example: String s1 = new String(“javaByDeveloper”);

Here objects will be created in heap and string constant pool. Here s1 refers to Heap.

## Memory Allocation

### String Constant Pool

If we are creating a string object, then with that String value JVM searches in the SCP area. If the value is not there then only it will create that object.

If an object already exists in SCP then existing object will be used instead of creating the new object.

i.e. Duplicate objects are not allowed.

Garbage collector doesn’t have the access to SCP area.

Objects created inside the SCP will be destroyed during the application deployment.

String s1 = “javaByDeveloper”;

String s2 = “javaByDeveloper”;

System.out.println(s1 == s2);

### Heap

Duplicate objects are allowed inside the heap area.

String s1 = new String(“javaByDeveloper”);

String s2 = new String(“javaByDeveloper”);

System.out.println(s1 == s2);

StringSample.java

In the above sample, we covered few scenarios related to strings. String method names are self-explanatory methods like length(),indexOf(), isEmpty (), contains() and lastIndexOf().You can try those methods on your own.

## StringBuffer:

StringBuffer also contains sequence of characters like strings but it is an **mutable** object.

i.e. length and content of the sequence can be changed through certain methods.

**Key Points:**

If a string value is going to modify frequently then we should use string buffer instead of String.

String buffer objects are thread safe.

The principal operations on a StringBuffer are the **append** and **insert** methods.

It is introduced in 1.5version.

StringBufferSample.java

The above program, demonstrate how to reverse a string using built in methods and appending new content to it.

## StringBuilder:

StringBuilder objects are like String objects, except that they can be modified. Internally, these objects are treated like variable-length arrays that contain a sequence of characters. At any point, the length and content of the sequence can be changed through method invocations.

StringBuilder object is more efficient than string.

It is introduced in 1.5version.

StringBuilderSample**.java**

Above program, demonstrates how to reverse a string using built in methods,append() , insert() and deleteCharAt() methods.

**Note**: There is also a **StringBuffer** class that is exactly the same as the **StringBuilder** class, except that it is thread-safe by virtue of having its methods synchronized.