Java String

Generally, a string is a sequence of characters. In C/C++ programming languages, it represents an array of characters, where the last character will be the null character (\0) that represents the end of the string.

But in Java, String is an object of String class that represents a string of characters. For example, “Pencil” is a string of 6 characters.

String class is used to create a string object. It is a predefined immutable class that is present in java.lang package.

But in Java, all classes are also considered as a data type. So, you can also consider a string as a data type. Immutable means it cannot be changed.

Basically, memory in Java is divided into three parts such as heap, stack, and String Pool. The string is so important in Java that it has a dedicated memory location. What string pool does it?

Whenever a string gets repeated it does not allocate the new memory for that string. It uses the same string by pointing a reference to it for saving memory.

Immutable String in Java

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.

```
package stringPrograms;
public class ImmutabilityTest
{
public static void main(String[] args)
{
   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.
 }
}
```

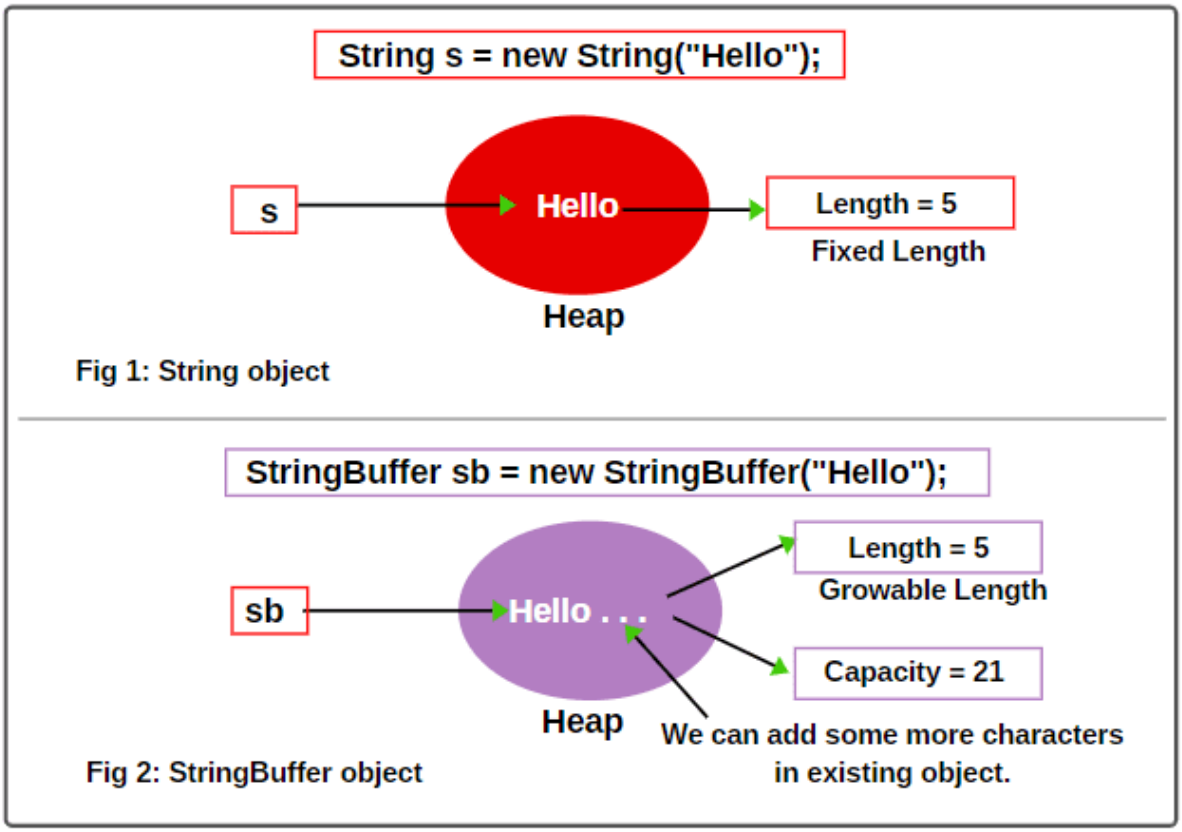
StringBuffer Class

StringBuffer in Java is a peer class of *String* that provides much of the functionality of strings. It provides more flexibility than String.

As we know that a string is fixed-length, immutable, and cannot be modified. That is, once we create a String object, we cannot change its content.

To overcome this, Java language introduced another class called StringBuffer.

Java StringBuffer represents a growable nature and mutable. That means that once we create a StringBuffer class object, we can perform any required changes in the object. i.e, its data can be modified.




```
package stringBufferPrograms;
public class CapacityTest {
public static void main(String[ ] args)
```

```
{
// Create a StringBuffer object.
  StringBuffer sb = new StringBuffer();
   int length = sb.length();
   int capacity = sb.capacity();

   System.out.println("Before adding any character:");
   System.out.println("Length = " +length); // length = 0
   System.out.println("Default initial capacity =  " +capacity); // Capacity = 16

// Adding 16 characters in the existing string buffer object.
   sb.append("abcdefghijklmnop");

   System.out.println("After adding 16 characters:");
   System.out.println("Length =  " +sb.length()); // now length is 16.
   System.out.println("Capacity =  " +sb.capacity()); // still capacity is 16.

// Now add the 17th character in the existing string buffer object.
   sb.append("q");

   System.out.println("After adding 17th character:");
   System.out.println("Length =  " +sb.length()); // length is 17
   System.out.println("Capacity =  " +sb.capacity()); // capacity is 34

// Add 17 characters to the existing object.
   sb.append("abcdefghijklmnopq"); // 17 characters. So, total = 34 characters.

// Adding 35th character.
   sb.append("r");

   System.out.println("After adding 35th character:");
   System.out.println("Length =  " +sb.length()); // length is 35
   System.out.println("Capacity =  " +sb.capacity()); // Capacity is 70
 }
}
```

```
package stringBufferPrograms;
public class CapacityTest2 {
public static void main(String[ ] args)
{
 StringBuffer sb = new StringBuffer("Tech");
  int length = sb.length();
  int capacity = sb.length() + 16;

  System.out.println(length); // 4
  System.out.println(capacity); // 20
  }
}
```

StringBuffer

In addition to be inherited from the Object class methods, StringBuffer class also provides some useful methods in Java. They are as:

- append()
- capacity()
- charAt()
- delete()
- ensureCapacity()
- getChars()
- indexOf()
- insert()
- length()
- reverse() and many more.

So, let's understand each *StringBuffer class* method one by one with example programs.

```
public class AppendDemo {
public static void main(String[] args)
{
  StringBuffer sb1 = new StringBuffer("Java");
  StringBuffer sb2 = new StringBuffer("Hello");
  StringBuffer sb3 = sb1.append(" Technology");
  StringBuffer sb4 = sb2.append(12345);
  System.out.println(sb3);
  System.out.println(sb4);
 }
}
```

```
public class LengthCapDemo {
public static void main(String[] args)
{
  StringBuffer sb = new StringBuffer("Java Technology");
  System.out.println("Original string: " +sb);
  int lengthSB = sb.length();
  int capacitySB = sb.capacity();
  System.out.println("Length of StringBuffer: " +lengthSB);
  System.out.println("Capacity of StringBuffer: " +capacitySB);
 }
}
```

```
public class LengthCapDemo {
public static void main(String[] args)
{
  StringBuffer sb = new StringBuffer("Hello World!");
  System.out.println("Original string: " +sb);
  int length = sb.length();
  int capacity = sb.capacity();
  System.out.println("Length: " +length);
  System.out.println("Capacity: " +capacity);

 sb.ensureCapacity(40);
 System.out.println("Now, capacity: " +sb.capacity());

 sb.setLength(15);
 System.out.println("Now, length: " +sb.length());
 }
}
```

```
ublic class StringBufferDemo {
public static void main(String[] args)
{
   StringBuffer sb = new StringBuffer("Live with");
   System.out.println("Buffer before: " + sb);
   System.out.println("charAt(1) before: " + sb.charAt(1));
   sb.setCharAt(1, 'o');
   sb.setLength(4);
   System.out.println("Buffer after setting length: " + sb);
   System.out.println("charAt(1) after = " + sb.charAt(1));
 }
}
```

```
public class StringBufferDemo {
public static void main(String[] args)
{
   StringBuffer sb = new StringBuffer("I love Java Programming");
   System.out.println("Original StringBuffer: " + sb);
   System.out.println("Character at index 7: " +sb.charAt(7));

   System.out.println("Unicode code point at index 7: " +sb.codePointAt(7));
   System.out.println("Unicode code point before index 7: " +sb.codePointBefore(7));
   System.out.println("Code points between indices 2 and 7: " +sb.codePointCount(2,
 7));
 }
}
```

```
public class StringBufferDemo {
public static void main(String[] args)
{
  StringBuffer sb = new StringBuffer("Java Programming");
```

```
  System.out.println("Original StringBuffer: " + sb);

   char[ ] c = new char[9];
    sb.getChars(0, 8, c, 0);
   System.out.println("Contents of character array:");
   for(int i = 0; i < c.length; i++) {
  System.out.print(c[i] + " ");
   }
 }
}
```

```
public class InsertDemo {
public static void main(String[] args)
{
   StringBuffer sb = new StringBuffer("I Java Programming!");
   System.out.println("Original StringBuffer: " + sb);
    sb.insert(2, "Like ");
   System.out.println("New StringBuffer: " +sb);
 }
}
```

```
public class DeleteDemo {
public static void main(String[] args)
{
  StringBuffer sb = new StringBuffer("I love mango");
  System.out.println("Original StringBuffer: " + sb);
   sb.delete(0, 7);
  System.out.println("After delete: " + sb);
   sb.deleteCharAt(1);
  System.out.println("After deleteCharAt: " + sb);
 }
}
```

```
public class ReplaceDemo {
public static void main(String[] args)
{
   StringBuffer sb = new StringBuffer("I hate you");
   System.out.println("Original StringBuffer: " + sb);
    sb.replace(2, 6, "love");
   System.out.println("After replace: " +sb);
  }
}
```

```
public class SearchDemo {
public static void main(String[] args)
{
  StringBuffer sb = new StringBuffer("I am Java Programmer");
  System.out.println("Original StringBuffer: " + sb);
```

```
   int i;
    i = sb.indexOf("Java");
    System.out.println("First index: " + i);
    i = sb.lastIndexOf("Am");
    System.out.println("Last index: " + i);
  }
}
```

```
public class ReverseDemo {
public static void main(String[] args)
{
  StringBuffer sb = new StringBuffer("ABCDEFGH");
  System.out.println("Original StringBuffer: " + sb);
    sb.reverse();
    System.out.println("New StringBuffer: " +sb);
 }
}
```

```
public class SubstringDemo {
public static void main(String[] args)
{
  StringBuffer sb = new StringBuffer("PQRSTUVWXYZ");
  System.out.println("Original StringBuffer: " + sb);
  String s1 = sb.substring(3);

  System.out.println("Substring: " +s1);
   String s2 = s1.substring(2, 5);
   System.out.println("New Substring: " +s2);
 }
}
```

```
public class ToStringDemo {
public static void main(String[] args)
{
 StringBuffer sb = new StringBuffer("PQRSTUVWXYZ");
 System.out.println("Original StringBuffer: " + sb);
 String s = sb.substring(2);
 String s2 = s.toString();
 System.out.println("String representation of StringBuffer: " +s2);
 }
}
```

```
public class tringBufferDemo {
public static void main(String[] args)
{
  StringBuffer sb = new StringBuffer("PQRST");
  System.out.println("Original StrinBuffer: " +sb);
  System.out.println("Original capacity: " +sb.capacity());
```

```
   sb.trimToSize();
   System.out.println("New capacity: " +sb.capacity());
   System.out.println("Runtime class: " +sb.getClass());
   System.out.println("Hashcode value: " +sb.hashCode());
  }
}
```