

## Length Variable vs Length() Method

### Length Variable:

1. `length` is a **final variable** applicable for **arrays**.
2. With the help of **length variable**, we can obtain the **size of the array**.

```
char ch[] = {'h', 'e', 'l', 'l', 'o'};  
System.out.println(ch); // hello  
System.out.println(ch.length); // 5
```

### Length Method()

1. `length()` method is a **final method** which is applicable for **string objects**.
2. `length()` method returns the number of **characters presents** in the **string**.

```
String s1 = "hello";  
System.out.println(s1); // hello  
System.out.println(s1.length()); // 5
```

## concat():

This method combines both the strings

```
public String concat(String str) {  
}
```

```
String s1 = "Java";  
String s2 = "Python";
```

```
System.out.println(s1.concat(s2)); // JavaPython
```

```
String s3 = "JavaScript";  
System.out.println(s3.concat(s1)); // JavaScriptJava
```

```
String s4 = "Spring";  
String s5 = "Django";  
String s6 = "React Js";
```

```
System.out.println(s4.concat(s5).concat(s6)); // SpringDjangoReact Js
```

```
String s7 = "MySQL";  
System.out.println(s7.concat("Oracle")); // MySQLOracle
```

## String Immutable:

String Objects are immutable once we create modification is not possible

Every time we are creating new objects s1, s2, s3 they are stored in SCP area but original string is not changing

```
String s1 = "Java";
```

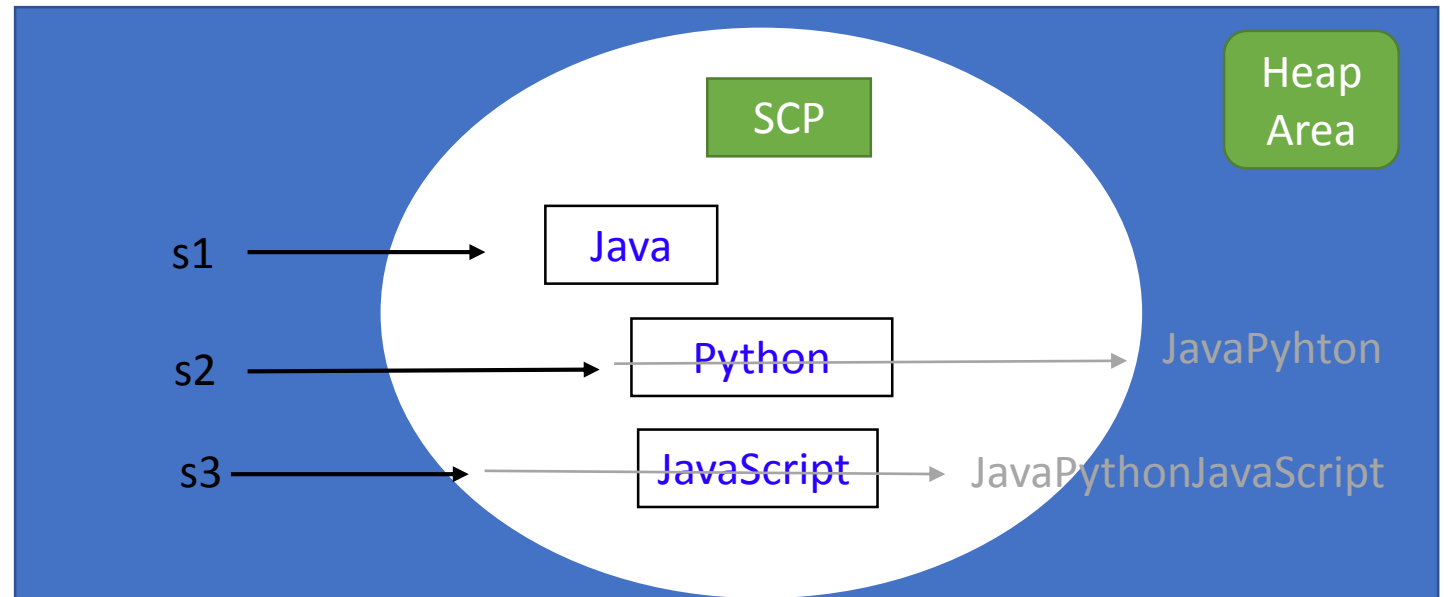
```
String s2 = s1.concat("Python");
```

```
String s3 = s2.concat("JavaScript");
```

```
System.out.println(s1); // Java
```

```
System.out.println(s2); // JavaPython
```

```
System.out.println(s3); // JavaPythonJavaScript
```



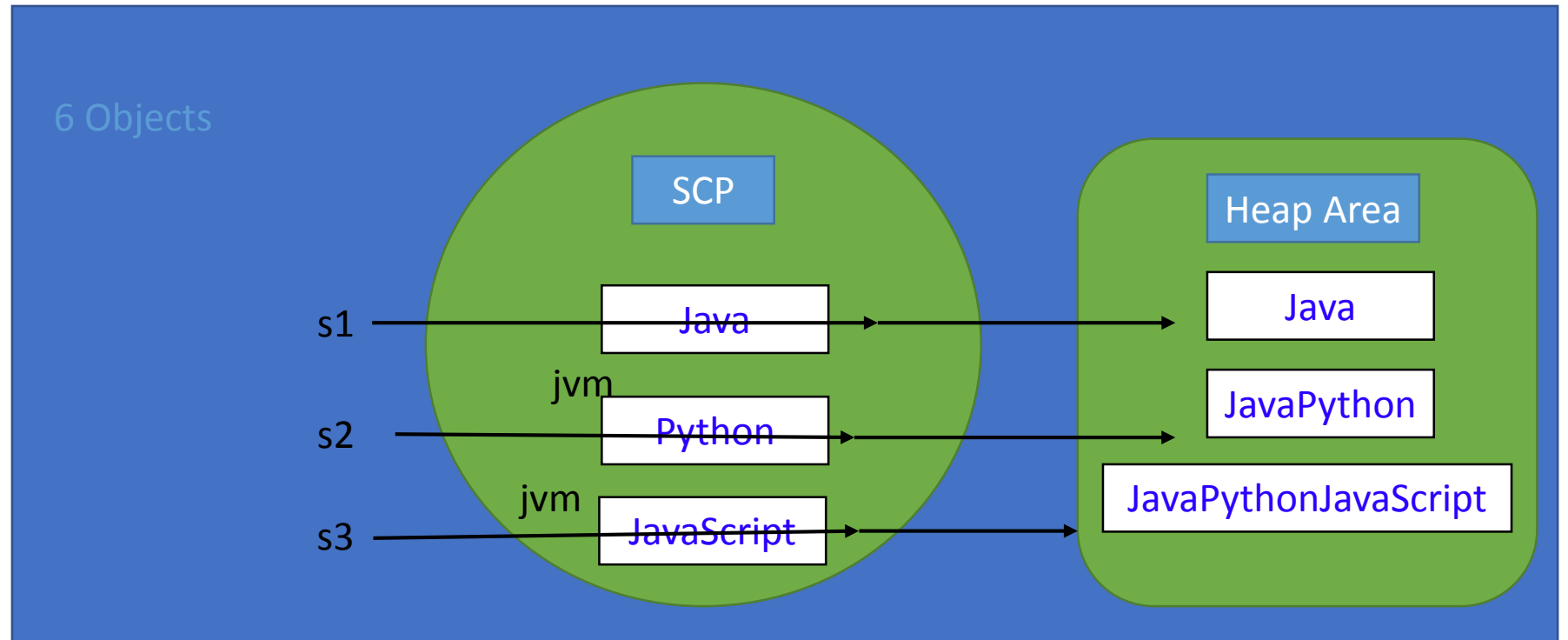
### String Immutable:

```
String s1 = new String("Java");  
String s2 = s1.concat("Python");  
String s3 = s2.concat("JavaScript");
```

```
System.out.println(s1); // Java
```

```
System.out.println(s2); // JavaPython
```

```
System.out.println(s3); // JavaPythonJavaScript
```



## `equals()`:

It compares the content of strings, if content is same returns true else it will return false

```
public boolean equals(Object anObject) {
```

```
}
```

```
String s1 = new String("Java");
```

```
String s2 = new String("Python");
```

```
String s3 = new String("JavaScript");
```

```
String s4 = new String("Java");
```

```
System.out.println(s1.equals(s2)); // false
```

```
System.out.println(s2.equals(s1)); // false
```

```
System.out.println(s3.equals(s2)); // false
```

```
System.out.println(s4.equals(s1)); // true
```

Diff bw == and .equals() method

== is used for refrence comparision

.equals() method for content comparision

```
int a = 10;  
int b = 20;  
int c = 10;
```

```
// reference comparison  
System.out.println(a == b); // false  
System.out.println(a == c); // true
```

```
String s1 = new String("Java");  
String s2 = new String("Python");  
String s3 = new String("Java");
```

```
// here object reference are different  
bcoz of objects  
System.out.println(s1 == s2); // false  
System.out.println(s1 == s3); // false
```

Stack Area

```
a = 10;  
b = 20;  
c = 10;
```

```
s1 → Java  
s2 → Python  
s3 → Java  
aa → A();  
aa → A();
```

Heap Area

SCP

Java  
Python  
Python

```
A aa = new A();  
A aaa = new A();  
System.out.println(aa==aaa); // false
```

Diff bw == and .equals() method

//here object class equals() method is called so its false, it works like == operator

```
B b1 = new B();
```

```
B b2 = new B();
```

```
System.out.println(b1.equals(b2)); //false
```

```
String s1 = new String("Java");
```

```
String s2 = new String("Python");
```

```
String s3 = new String("Java");
```

// content comparison for string objects, if data is there then it will return true or else false

//here string class equals() method is called so its true

```
System.out.println(s1.equals(s2)); //false
```

```
System.out.println(s1.equals(s3)); //true
```

## equalsIgnoreCase()

This method is used to compare the two strings irrespective of the content (lower and upper case)

```
public boolean equalsIgnoreCase(String anotherString) {  
  
}
```

```
String s1 = new String("Java");  
String s2 = new String("java");
```

```
System.out.println(s1.equals(s2)); // false  
System.out.println(s1.equalsIgnoreCase(s2)); // true
```



## compareTo()

The comparison is based on the Unicode value of each character in the strings.

```
public int compareTo(String anotherString) {  
}
```

```
String s1 = new String("A");  
String s2 = new String("B");  
String s3 = new String("A");  
System.out.println(s1.compareTo(s2)); // s1<s2 returns negative  
System.out.println(s1.compareTo(s3)); // s1 == s2 returns 0  
System.out.println(s2.compareTo(s3)); // s2>s3 returns positive
```

## compareToIgnoreCase()

```
String s1 = new String("A");
```

```
String s2 = new String("a");
```

```
String s3 = new String("B");
```

```
System.out.println(s1.compareToIgnoreCase(s2)); // s1 == s2 returns 0 irrespective of case sensitive
```

```
System.out.println(s1.compareTo(s2)); //65-97 = -32 //-32
```

```
System.out.println(s1.compareToIgnoreCase(s3)); //-1
```

## Object class equals() method and String class equals() method

//here object class equals() method is called so its false, it works like == operator

```
B b1 = new B();
```

```
B b2 = new B();
```

```
System.out.println(b1.equals(b2)); //false
```

//here string class equals() method is called so its true

// content comparison for string objects, if data is there then it will return true or else false

```
String s1 = new String("Java");
```

```
String s2 = new String("Python");
```

```
String s3 = new String("Java");
```

```
System.out.println(s1.equals(s2)); //false
```

```
System.out.println(s1.equals(s3)); //true
```

**startsWith()** method checks if this string starts with the given prefix.  
It returns true if this string starts with the given prefix; else returns false.

```
public boolean startsWith(String prefix) {  
}
```

```
String s1 = new String("Java and Python");  
System.out.println(s1.startsWith("J")); // true  
System.out.println(s1.startsWith("P")); // false
```

**endsWith()** method checks if this string ends with a given suffix.  
It returns true if this string ends with the given suffix; else returns false.

```
public boolean endsWith(String suffix) {  
}
```

```
System.out.println(s1.endsWith("Python")); // true  
System.out.println(s1.endsWith("Java")); // false
```

**contains()** method searches the sequence of characters in this string.  
It returns true if the sequence of char values is found in this string otherwise returns false.

```
public boolean contains(CharSequence s) {  
}
```

```
String s1 = new String("Java and Python");  
System.out.println(s1.contains("and")); // true  
System.out.println(s1.contains("is")); // false
```

**charAt()** method returns a char value at the given index number

```
public char charAt(int index) {
```

```
}
```

```
String s1 = new String("Programming");
```

```
System.out.println(s1.charAt(0)); // P
```

```
System.out.println(s1.charAt(4)); // r
```

```
System.out.println(s1.charAt(20)); // java.lang.StringIndexOutOfBoundsException
```

**replace()** method returns a string replacing all the old char to new char

```
public String replace(CharSequence target, CharSequence replacement) {  
}
```

```
String s1 = new String("Java");  
System.out.println(s1.replace("J", "j")); // java
```

```
String s2 = new String("Python");  
System.out.println(s2.replace("Python", "Java")); // Java
```

```
String s3 = new String("Java Programming Java Programs");  
System.out.println(s3.replace("Pro", "pro")); // Java programming Java programs  
System.out.println(s3.replaceAll("a", "A")); // JAvA ProgrAMming JAvA ProgrAms  
System.out.println(s3.replaceFirst("P", "p")); // Java programming Java Programs
```

**//replace the string from middle using index**

**public static void** main(String[] args) {

String s1 = **new** String("Hello Java and Python");

**char[]** ch = s1.toCharArray(); **//converts this string to a new char array**

ch[6] = 'j';

System.**out.println**(ch); **// Hello java and Python**

}



**lastIndexOf()** method returns the last index of the given character value or substring. If it is not found, it returns -1. The index counter starts from zero.

```
public int lastIndexOf(String str) {  
}
```

```
// We are having 3 "likes" but it searches from last index  
String s1 = "I like Java and I like JavaScript and I like Python";  
System.out.println(s1.lastIndexOf("like")); // 40  
System.out.println(s1.lastIndexOf("live")); // -1
```

## indexOf() method

```
public int indexOf(String str) {  
}
```

// We are having 3 "likes" but it searches from beginning index

```
String s2 = "I like Java and I like JavaScript";
```

```
System.out.println(s2.indexOf("like")); // 2--> Starts finding from beginning of statement
```

```
public int indexOf(String str, int fromIndex) {  
}
```

```
System.out.println(s2.indexOf("like", 7)); // 18
```

```
public String toUpperCase() {
```

```
}
```

```
String s = "Java"; // 0123
```

```
// toUpperCase
```

```
System.out.println(s.toUpperCase()); // JAVA
```

```
public String toLowerCase() {
```

```
}
```

```
// toLowerCase
```

```
System.out.println(s.toLowerCase()); // java
```

//remove character using substring() method

```
public String substring(int beginIndex) {
```

```
}
```

```
String s = "Java"; // 0123
```

```
System.out.println(s.substring(1));// ava
```

```
public String substring(int beginIndex, int endIndex) {
```

```
}
```

```
System.out.println(s.substring(1, 3));// av
```

**//delete the character from the string**

```
String s2 = new String("Hello Java and Python");
```

```
StringBuilder sb = new StringBuilder(s2);
```

```
sb.deleteCharAt(3);
```

```
System.out.println(sb); // Helo Java and Python
```