String:

1.String is an object typically representing sequence of characters

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

2.String are immutable once we create we cannot modify it

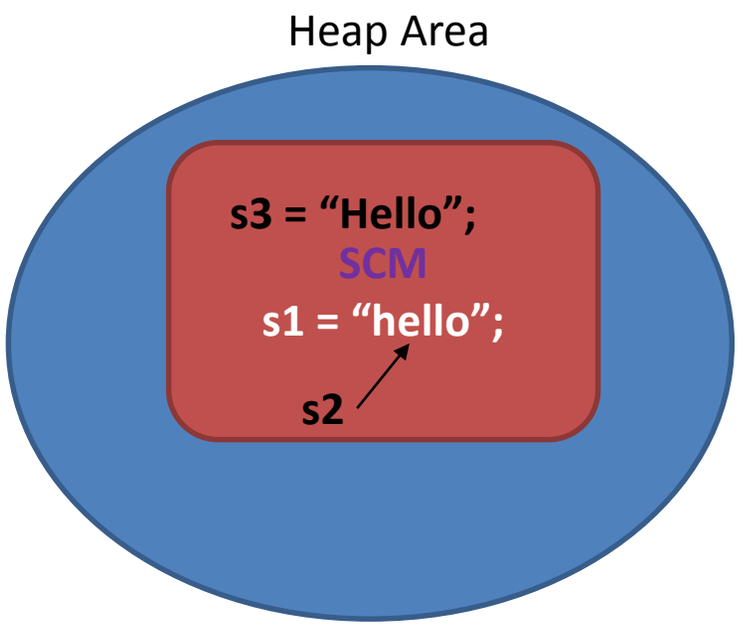
3.The **java.lang.String** class is used to create string object.

4.Strings are always in double quotes.

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

String Literals:

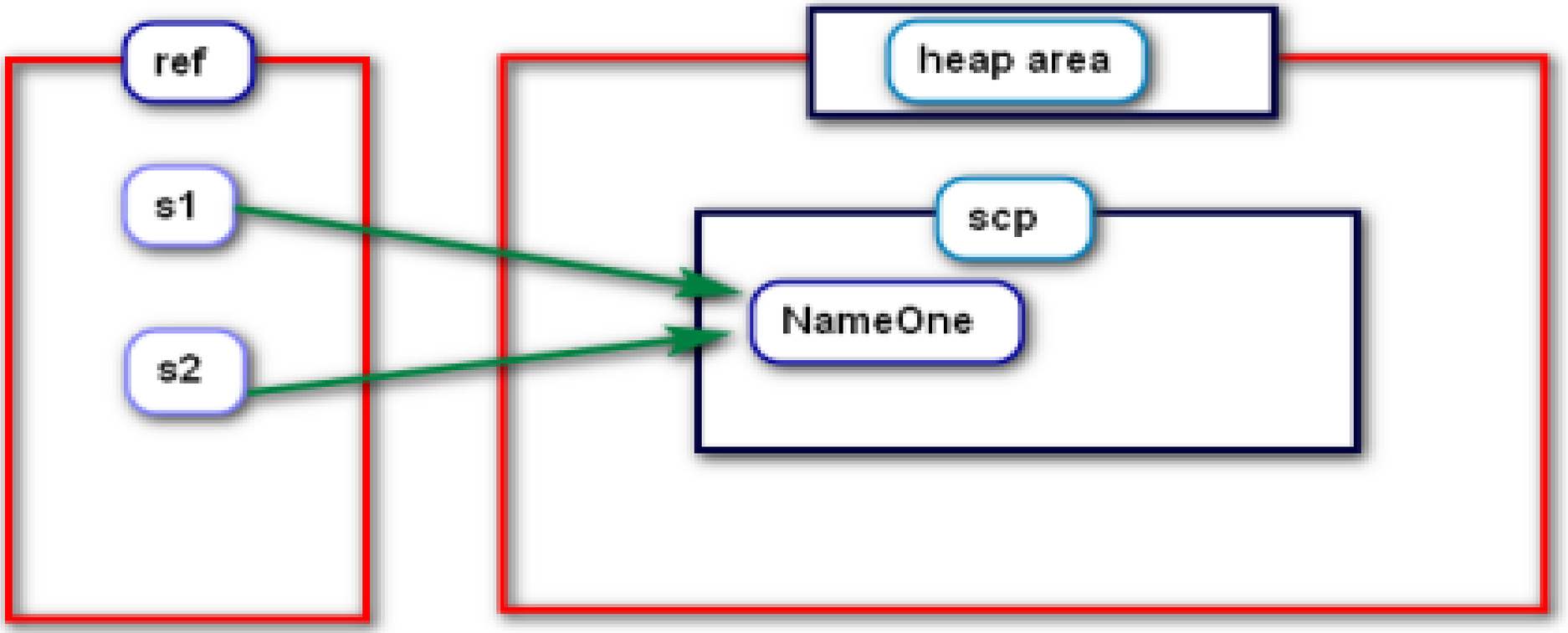
When we create a string in java like **String s1="hello";**
then an object will be created in string pool(hello) and
s1 will be pointing to hello.
Now if again we do **String s2="hello";**
then another object will not be created but **s2** will point to hello because **JVM will first check if the same object is present in string pool or not.**
If not present then only a new one is created else not.



Note:

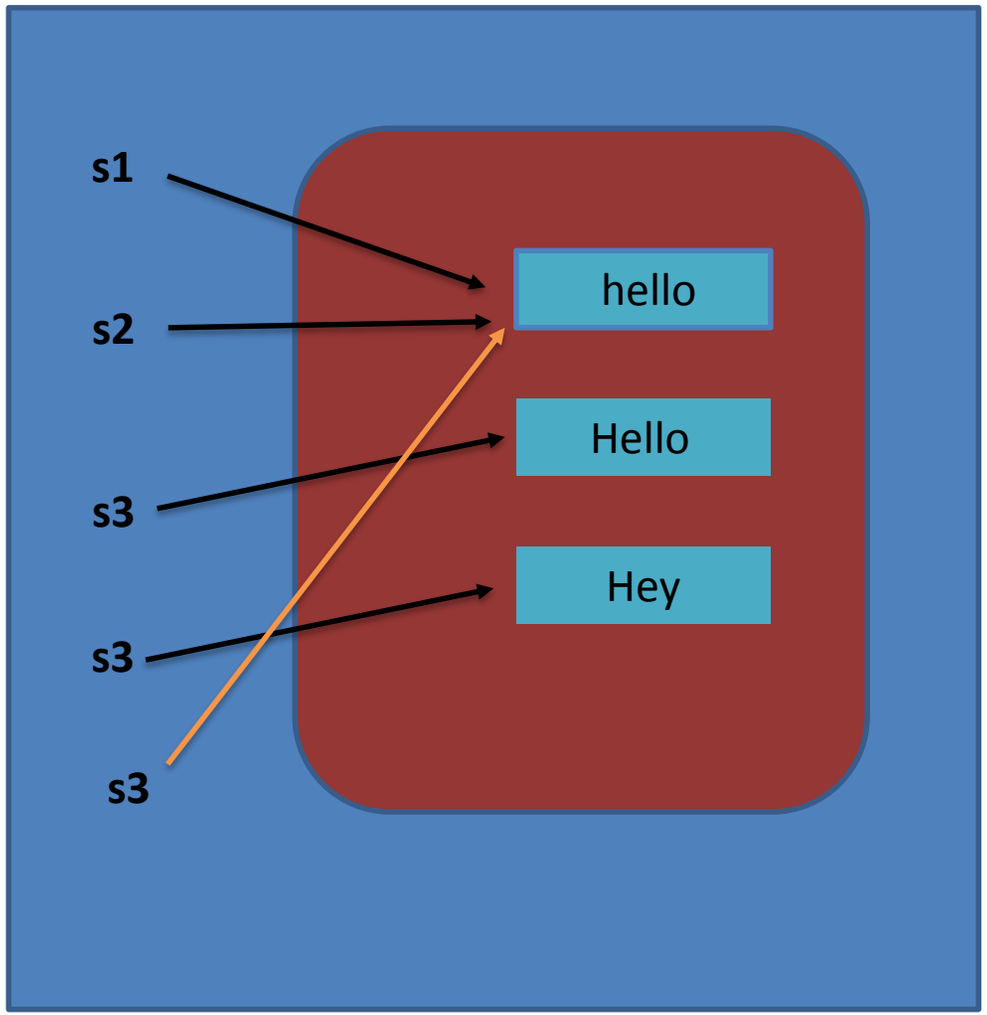
String Constant Pool is a memory in **Method area**
Suppose if any object is created in String Constant Pool area then that object is not eligible for **Garbage Collector**
The data will be cleaned once the execution is completed

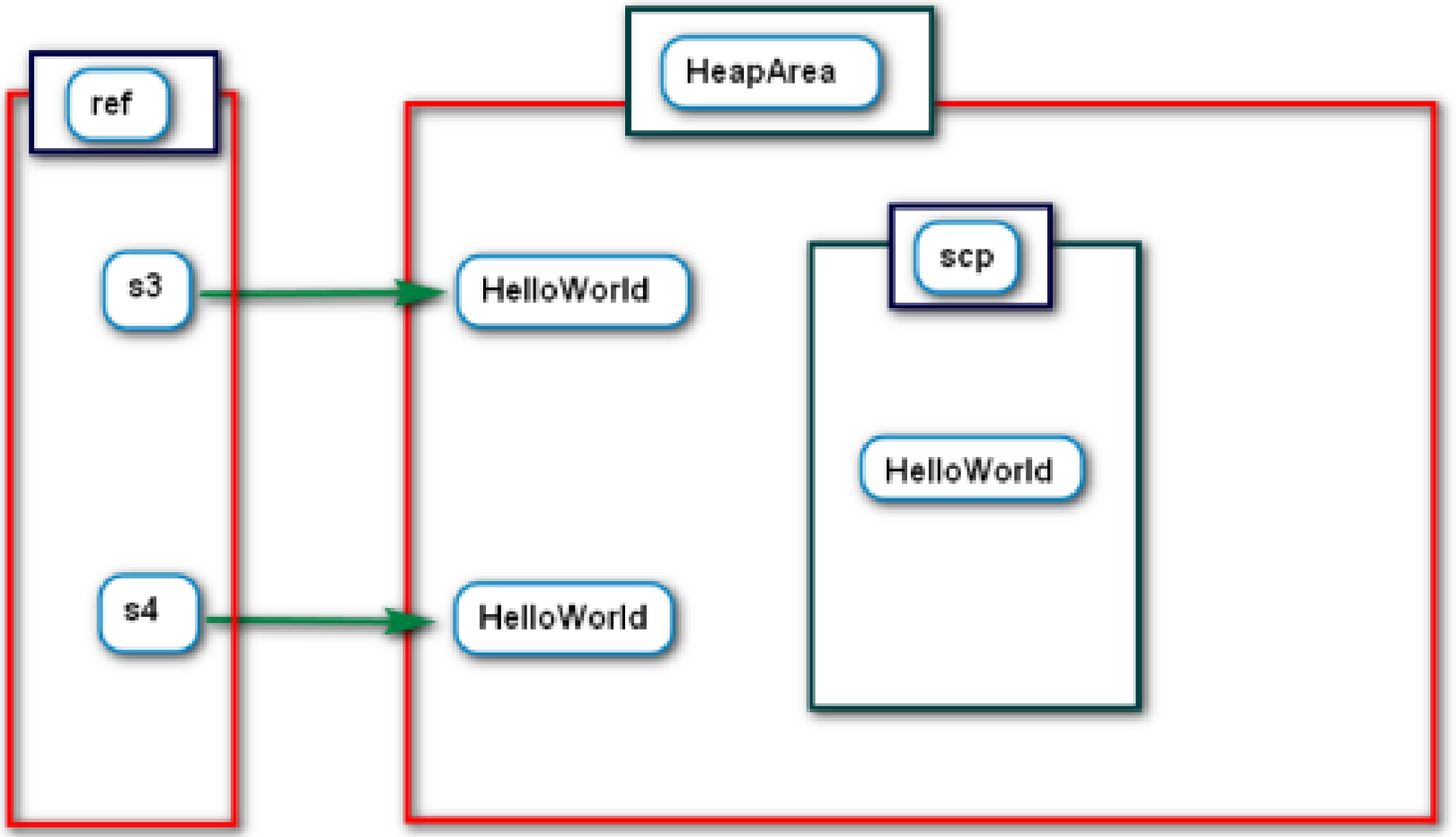
Note: In latest versions of java **SCP area is moved to Heap Memory**




```
String s1 = "NameOne"; //single object in scp
String s2 = "NameOne"; //single object in scp
System.out.println(s1 == s2); // true
```

```
// String Literals object is created in String constant pool area
String s1 = "hello";
System.out.println(s1); // hello

String s2 = "hello";
System.out.println(s2); // hello

String s3 = "Hello";
System.out.println(s3); // Hello

//Manipulating the String now points to diff memory
s3 = "Hey";
System.out.println(s3); // Hey
System.out.println(s1 == s3); // false

//Manipulating the String now points to diff memory
s3 = "hello";
System.out.println(s3); // hello
System.out.println(s1 == s3); // true

// reference comparison
System.out.println(s1 == s2); // true
System.out.println(s1 == s3); // true
```






```
String s3 = new String("HelloWorld"); // 2 object in heap area and scp
String s4 = new String("HelloWorld"); // 2 object in hepa area and scp
System.out.println(s3==s4); // false
```

String using new Keyword

```
String s3 = new String("HelloWorld");
System.out.println(s3); // HelloWorld
```

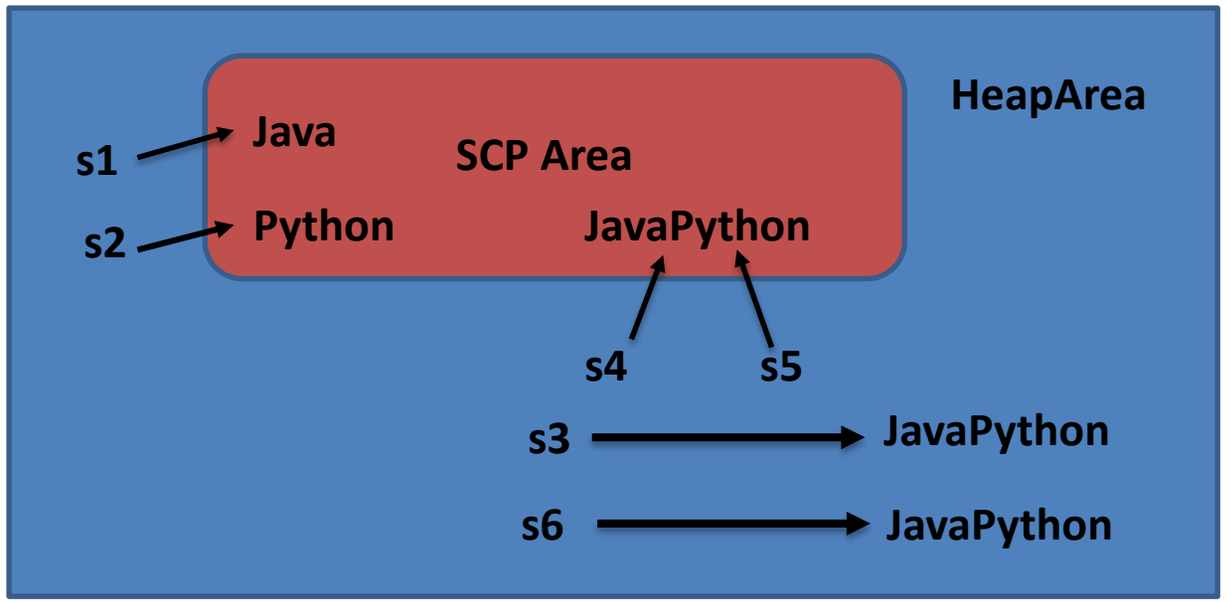
JVM will create a String object with object ref s3 in SCP area
If any String object is existed with same data in SCP area then JVM will not create new String Object

```
String s4 = new String("HelloWorld");
System.out.println(s4); // HelloWorld
```

Again if we create String object with object ref s4, new object is created in Heap Area
If any String objects are created in Heap Memory then that objects are available for Garbage Collector

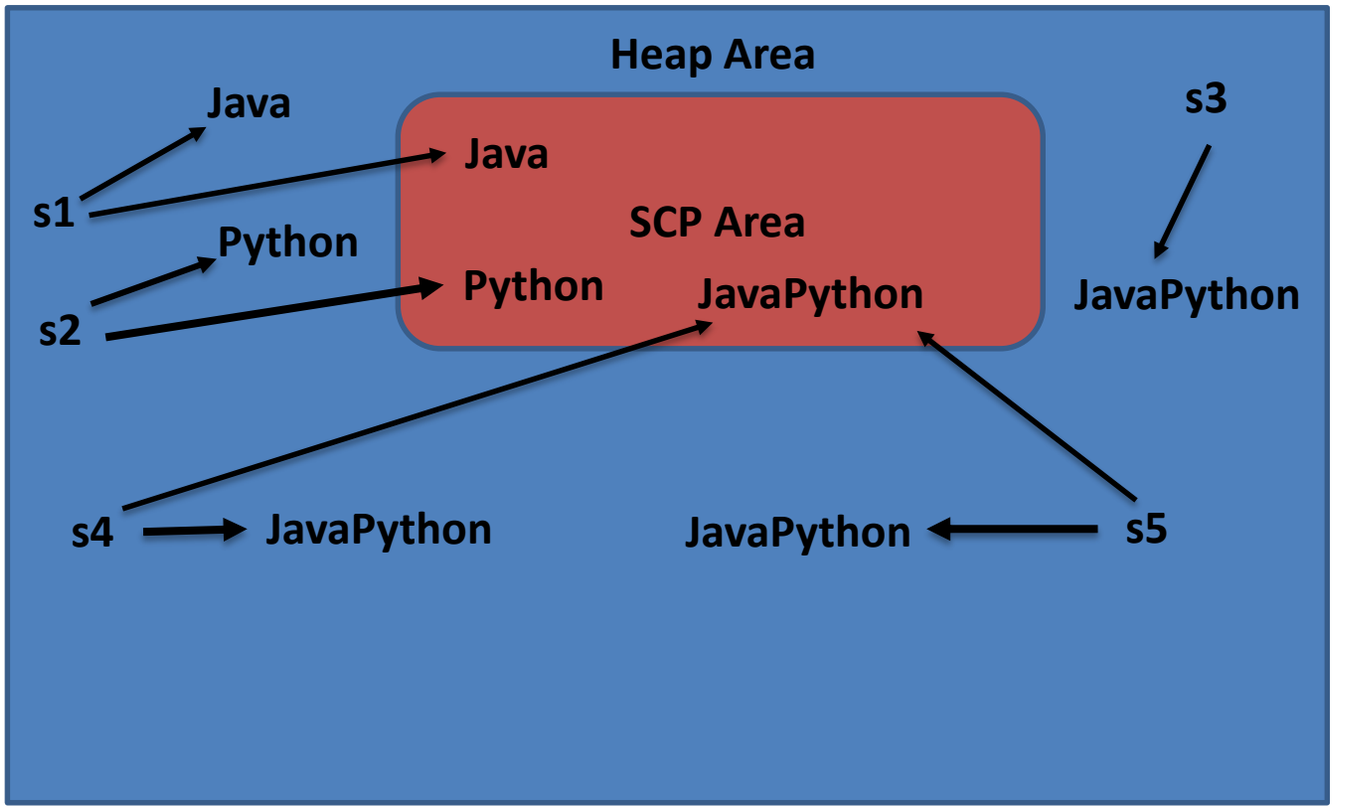
String is immutable, it will not modify orginal object

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

System.out.println(s1+s2); // JavaPython

String s3 = s1.concat(s2);
System.out.println(s3); // JavaPython

String s4 = "JavaPython";
System.out.println(s4); // JavaPython
System.out.println("String Concat Comparision: " + (s4==s3)); // String Concat Comparision: false

String s5 = "JavaPython";
System.out.println("String Compariosion: " + (s5==s4)); // String Compariosion: true

String s6 = s1.concat(s2);
System.out.println(s6);
System.out.println("String Concat Comparision: " + (s6==s3)); // String Concat Comparision: false
```



String is immutable, it will not modify orginal object

```
String s1 = new String("Java");
String s2 = new String("Python");
s1.concat(s2);
System.out.println(s1); // Java
System.out.println(s2); // Python

String s3 = s1.concat(s2);
System.out.println(s3); // JavaPython

String s4 = new String("JavaPython");
System.out.println(s4); // JavaPython
System.out.println(s3 == s4); // false

String s5 = new String("JavaPython");
System.out.println(s5); // JavaPython
System.out.println(s5==s4); // false
```



Object type and String type

```
public class Client {

public static void main(String[] args) {

//Object data is mutable
Products p1 = new Products(101, "Samsung");
System.out.println("Hashcode: " + p1); // com.dl.one.Products@626b2d4a

Products p2 = new Products(102, "Lg");
System.out.println("Hashcode: " + p2); // com.dl.one.Products@5e91993f

//String data is immutable
String s1 = new String("NameOne");
System.out.println("Immutable: " + s1); // Immutable: NameOne
System.out.println("Hashcode: " + s1.hashCode()); // Hashcode: -908377157

}
}
```

```
class Products {

int pId;
String pName;

public Products(int pId, String
pName) {
this.pId = pId;
this.pName = pName;
}
}
```

Converting byte data to String data using Constructors

String(byte[] bytes) is constructor can be used to get ASCII data

```
byte[] bytes = { 65, 66, 67, 68, 69 };
String s1 = new String(bytes);
System.out.println(s1); // ABCDE
```

Converting byte data to String data using Constructors

String(byte[] bytes, int offset, int length) is constructor can be used to get ASCII data and also it call the elements from given start index 1 and length is 3, so call next elements from index 1

```
byte[] bytes = { 65, 66, 67, 68, 69};
String s2 = new String(bytes, 1, 3);
System.out.println(s2); // BCD
```

Converting the data from char array to String:

public String(char value[])

```
char[] ch = {'A','B', 'C', 'D','E'};
String s1 = new String(ch);
System.out.println(s1); // ABCDE
```

public String(char value[], int offset, int count)

```
String s2 = new String(ch, 1, 3);
System.out.println(s2); // BCD
```

String Buffer and String Builder Constructors

```
//Thread Safety, Available Since 1.0, Mutable
public StringBuffer(String str) {
        super(str);
 }
```

```
StringBuffer stringBuffer = new StringBuffer("Hello Java");
System.out.println(stringBuffer); // Hello Java
```

```
//No Thread Safety, Available Since 1.5, Mutable
public StringBuilder(String str) {
        super(str);
}
```

```
StringBuilder stringBuilder = new StringBuilder("Hello Java");
System.out.println(stringBuilder); // Hello Java
```