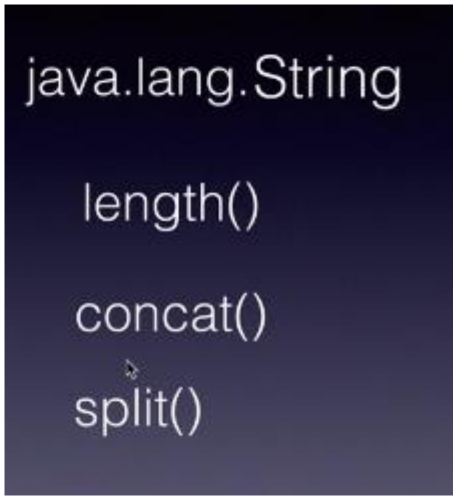
Strings

***Java.lang.String*:-**

String is used to represent group of characters or character array enclosed with in the double quotes.



Various ways of creating String in Java

There are two ways to create String object:

1. By string literal
2. By new keyword

---

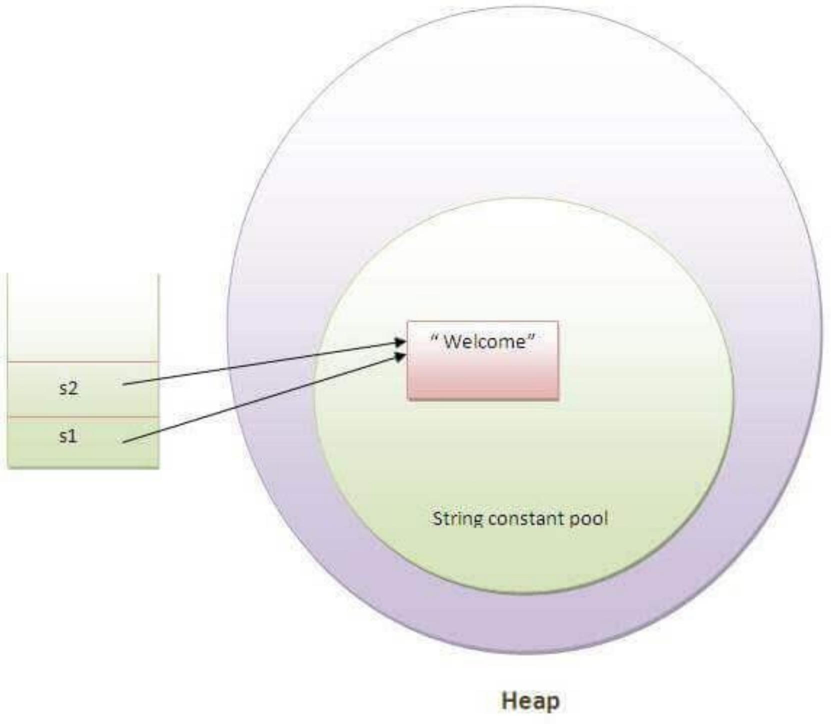
1) String Literal

Java String literal is created by using double quotes. For Example:

1. String s="welcome";

Each time you create a string literal, the JVM checks the "**string constant pool**" first. If the string already exists in the pool, a reference to the pooled instance is returned. If the string doesn't exist in the pool, a **new string instance is created** and placed in the pool. For example:

1. String s1="Welcome";
2. String s2="Welcome";//It doesn't create a new instance



In the above example, only one object will be created. Firstly, JVM will not find any string object with the value "Welcome" in string constant pool, that is why it will create a new object. After that it will find the string with the value "Welcome" in the pool, it will not create a new object but will return the reference to the same instance.

Note: String objects are stored in a special memory area known as the "**string constant pool**".

Why Java uses the concept of String literal?

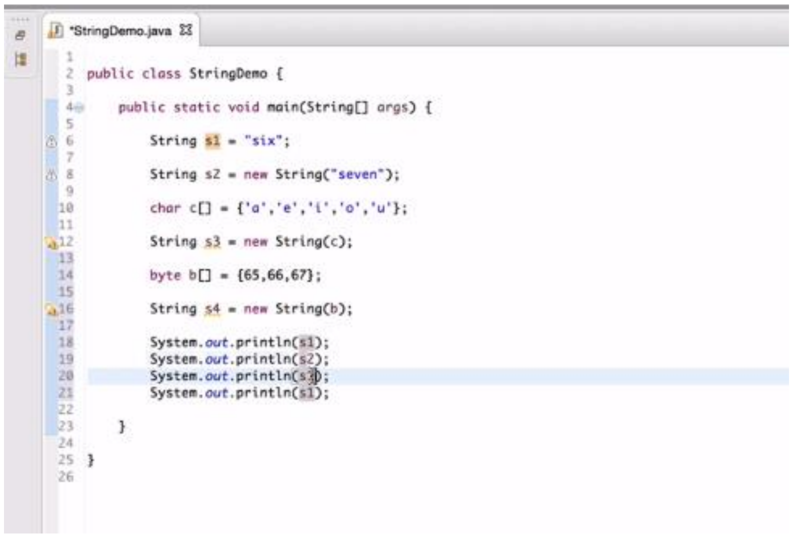
To make **Java more memory efficient** (because no new objects are created if it exists already in the string constant pool).

---

2) By **new keyword**

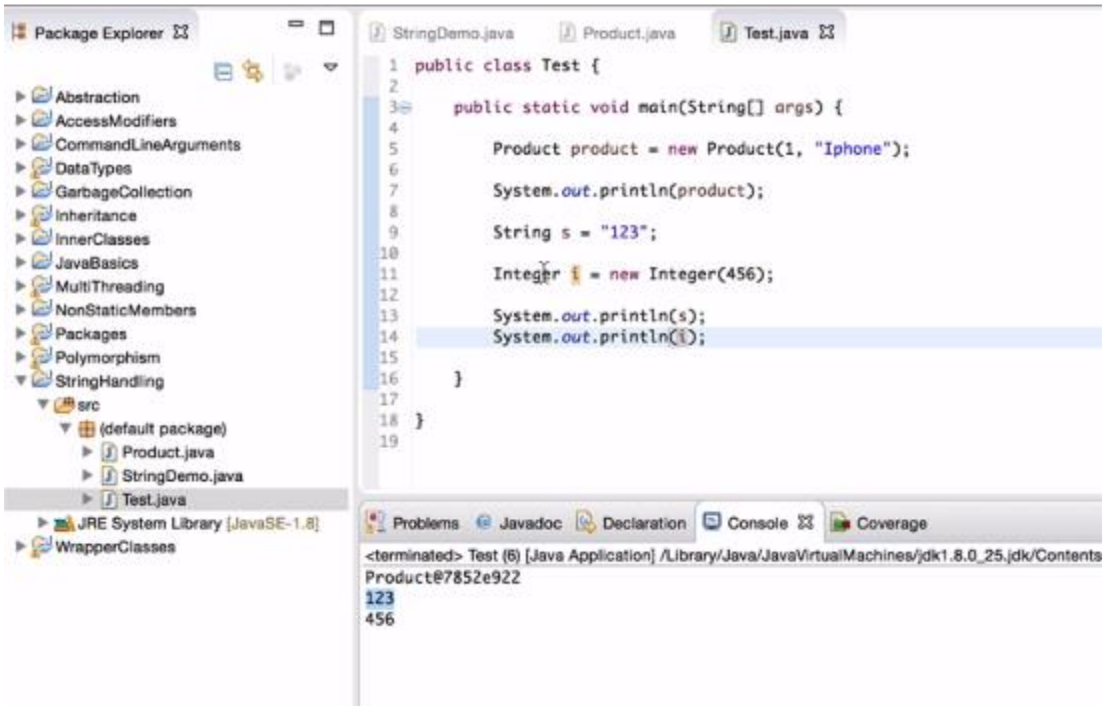
1. String s=new String("Welcome");

In such case, JVM will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the string constant pool. The variable s will refer to the object in a heap (non-pool).



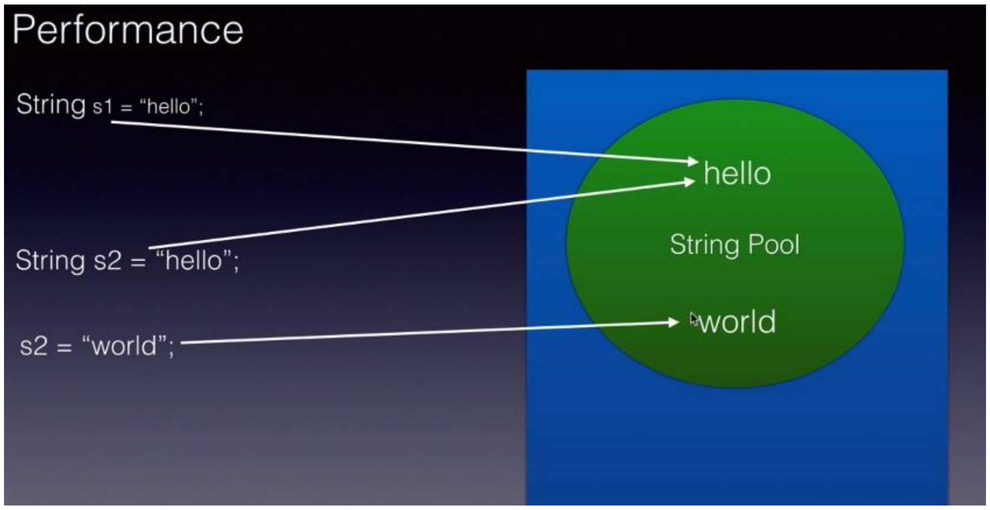
*StringDemo.java

```java

public class StringDemo {

    public static void main(String[] args) {

        String s1 = "six";

        String s2 = new String("seven");

        char c[] = {'a','e','i','o','u'};

        String s3 = new String(c);

        byte b[] = {65,66,67};

        String s4 = new String(b);

        System.out.println(s1);
        System.out.println(s2);
        System.out.println(s3);
        System.out.println(s1);

    }

}
```

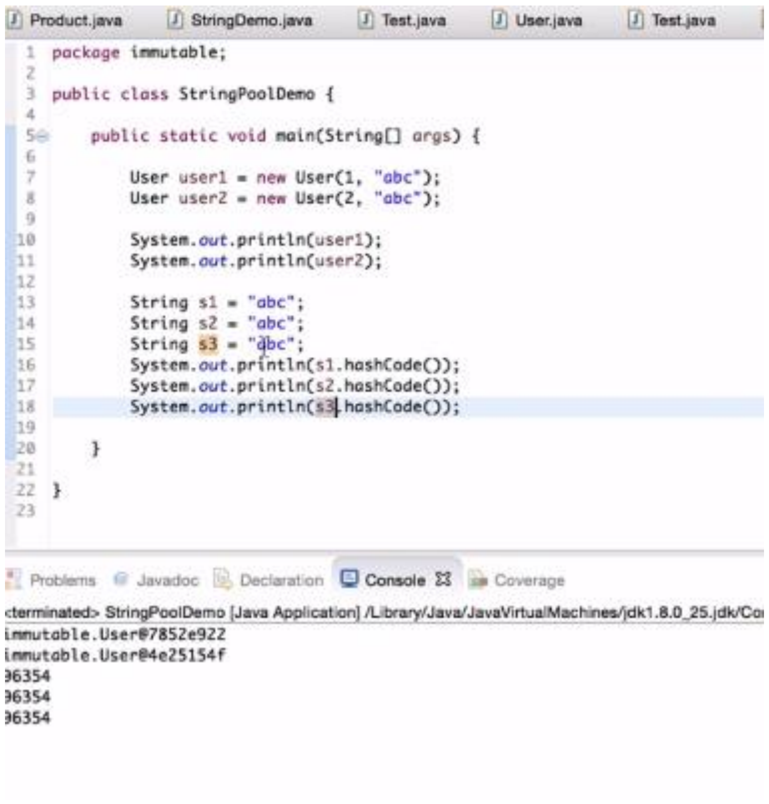


```java
public class Test {

    public static void main(String[] args) {

        Product product = new Product(1, "Iphone");

        System.out.println(product);

        String s = "123";

        Integer i = new Integer(456);

        System.out.println(s);
        System.out.println(i);

    }

}
```

```
<terminated> Test (6) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents
Product@7852e922
123
456
```



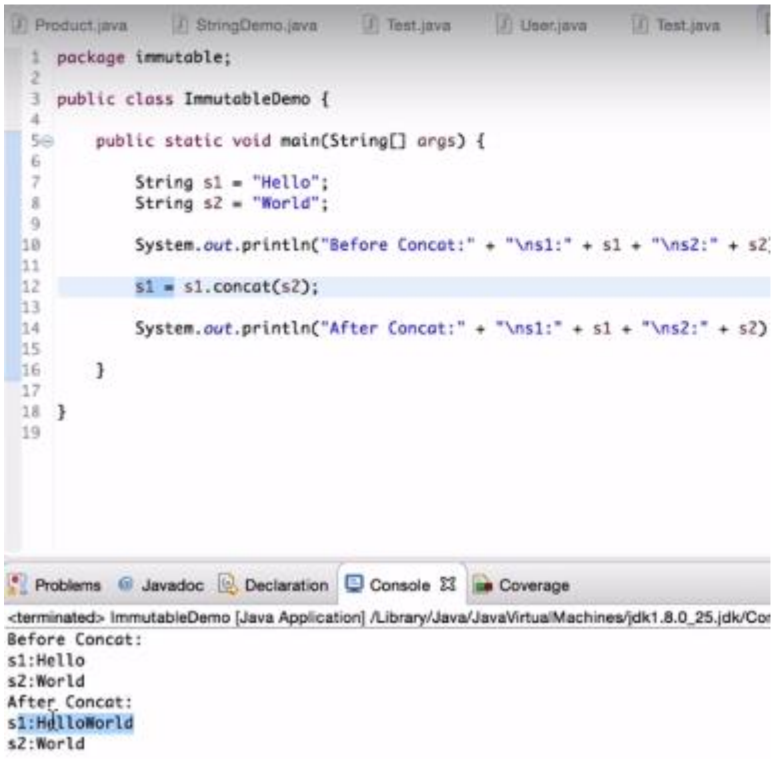
Example program to show both are of same locations



```java
package immutable;

public class StringPoolDemo {

    public static void main(String[] args) {

        User user1 = new User(1, "abc");
        User user2 = new User(2, "abc");

        System.out.println(user1);
        System.out.println(user2);

        String s1 = "abc";
        String s2 = "abc";
        String s3 = "abc";
        System.out.println(s1.hashCode());
        System.out.println(s2.hashCode());
        System.out.println(s3.hashCode());

    }

}
```

```
<terminated> StringPoolDemo [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Co
immutable.User@7852e922
immutable.User@4e25154f
96354
96354
96354
```

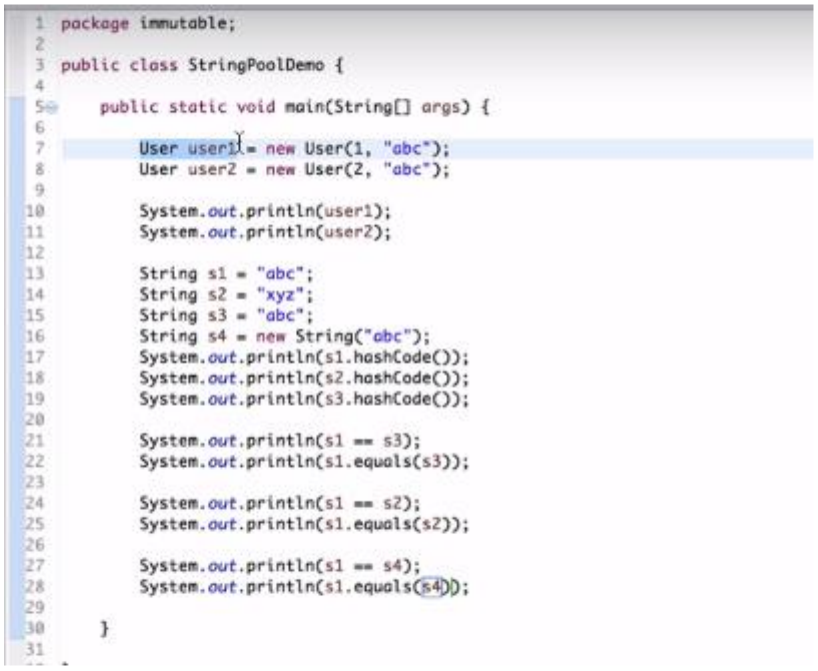
Concat Operation



```java
package immutable;

public class ImmutableDemo {

    public static void main(String[] args) {

        String s1 = "Hello";
        String s2 = "World";

        System.out.println("Before Concat:" + "\ns1:" + s1 + "\ns2:" + s2

        s1 = s1.concat(s2);

        System.out.println("After Concat:" + "\ns1:" + s1 + "\ns2:" + s2)

    }

}
```

```
<terminated> ImmutableDemo [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Co
Before Concat:
s1:Hello
s2:World
After Concat:
s1:HelloWorld
s2:World
```

Difference between (==) and equals method in java ?

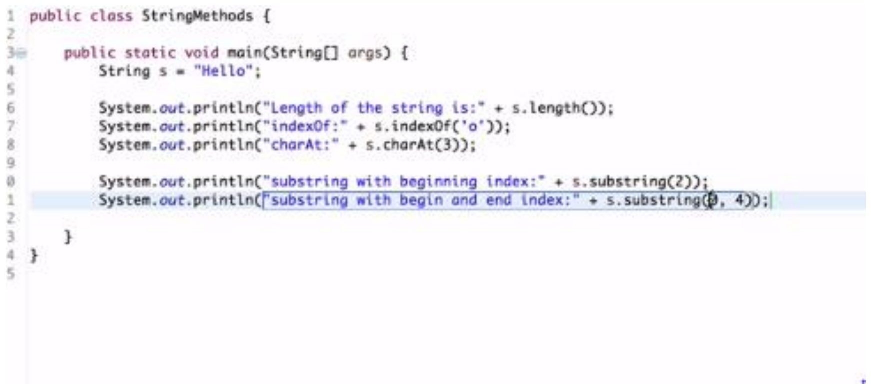


```java
package immutable;

public class StringPoolDemo {

    public static void main(String[] args) {

        User user1 = new User(1, "abc");
        User user2 = new User(2, "abc");

        System.out.println(user1);
        System.out.println(user2);

        String s1 = "abc";
        String s2 = "xyz";
        String s3 = "abc";
        String s4 = new String("abc");
        System.out.println(s1.hashCode());
        System.out.println(s2.hashCode());
        System.out.println(s3.hashCode());

        System.out.println(s1 == s3);
        System.out.println(s1.equals(s3));

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

        System.out.println(s1 == s4);
        System.out.println(s1.equals(s4));

    }
```

Utility Methods in String class ...

a)indexOf

b)charAt



```java
public class StringMethods {

    public static void main(String[] args) {
        String s = "Hello";

        System.out.println("Length of the string is:" + s.length());
        System.out.println("indexOf:" + s.indexOf('o'));
        System.out.println("charAt:" + s.charAt(3));

        System.out.println("substring with beginning index:" + s.substring(2));
        System.out.println("substring with begin and end index:" + s.substring(0, 4));

    }
}
```

c)substring() {overloaded methods}

d)replace()

e)toUpperCase()

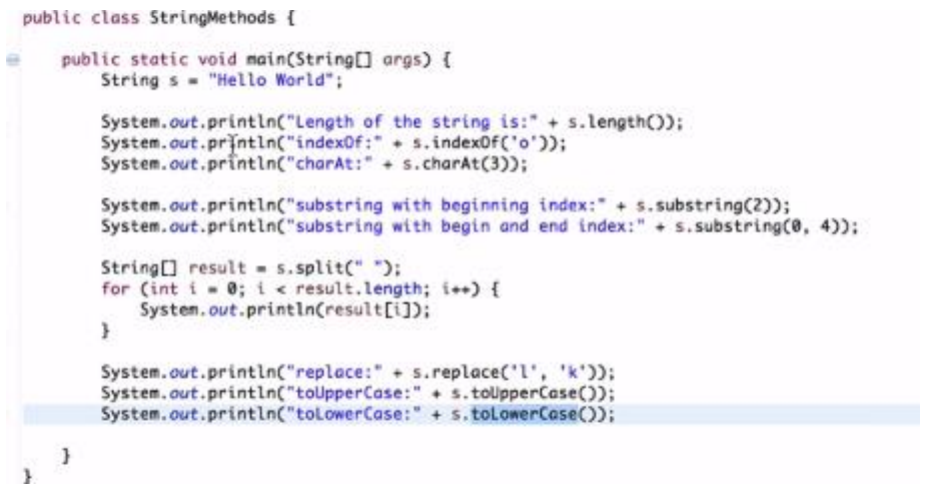
f)toLowerCase()

g)split();

h)replace();

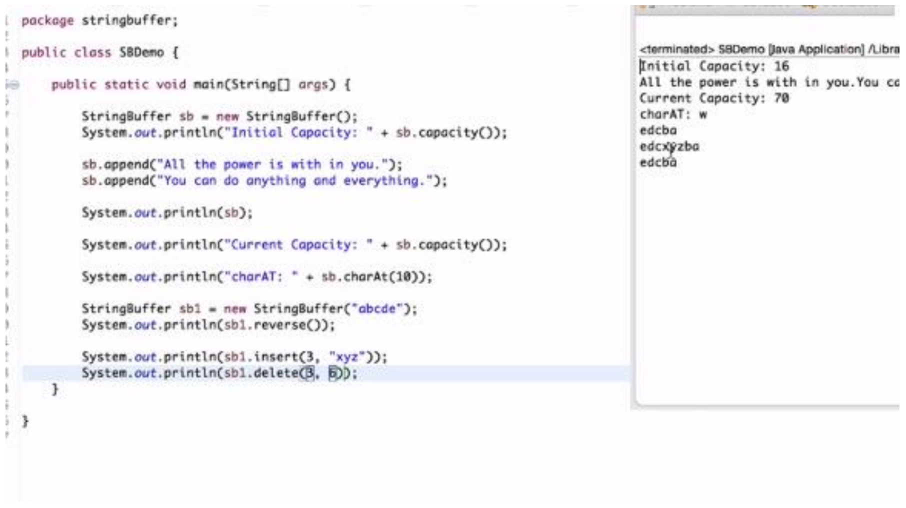
i)equals

j)equalsIgnoreCase()



```java
public class StringMethods {

    public static void main(String[] args) {
        String s = "Hello World";

        System.out.println("Length of the string is:" + s.length());
        System.out.println("indexOf:" + s.indexOf('o'));
        System.out.println("charAt:" + s.charAt(3));

        System.out.println("substring with beginning index:" + s.substring(2));
        System.out.println("substring with begin and end index:" + s.substring(0, 4));

        String[] result = s.split(" ");
        for (int i = 0; i < result.length; i++) {
            System.out.println(result[i]);
        }

        System.out.println("replace:" + s.replace('l', 'k'));
        System.out.println("toUpperCase:" + s.toUpperCase());
        System.out.println("toLowerCase:" + s.toLowerCase());

    }
}
```

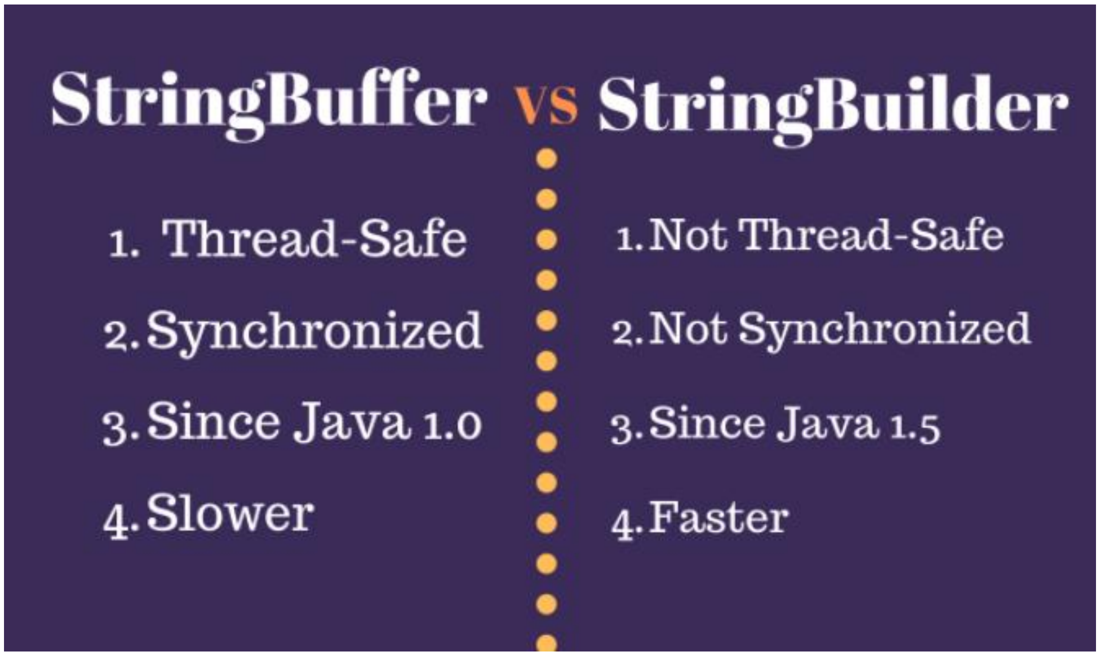
String Buffer class In Java ? (Mutable)



```java
package stringbuffer;

public class SBDemo {

    public static void main(String[] args) {

        StringBuffer sb = new StringBuffer();
        System.out.println("Initial Capacity: " + sb.capacity());

        sb.append("All the power is with in you.");
        sb.append("You can do anything and everything.");

        System.out.println(sb);

        System.out.println("Current Capacity: " + sb.capacity());

        System.out.println("charAT: " + sb.charAt(10));

        StringBuffer sb1 = new StringBuffer("abcde");
        System.out.println(sb1.reverse());

        System.out.println(sb1.insert(3, "xyz"));
        System.out.println(sb1.delete(3, 6));
    }

}
```

```
<terminated> SBDemo [Java Application] /Libra
Initial Capacity: 16
All the power is with in you.You c
Current Capacity: 70
charAT: w
edcba
edcxyzba
edcba
```

String Builder vs String Buffer ?



StringBuffer	StringBuilder
1. Thread-Safe	1. Not Thread-Safe
2. Synchronized	2. Not Synchronized
3. Since Java 1.0	3. Since Java 1.5
4. Slower	4. Faster