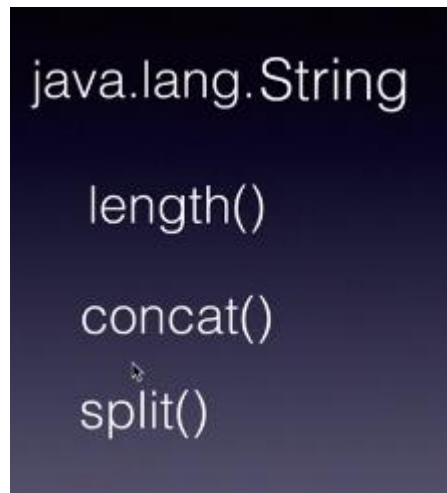


Strings

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

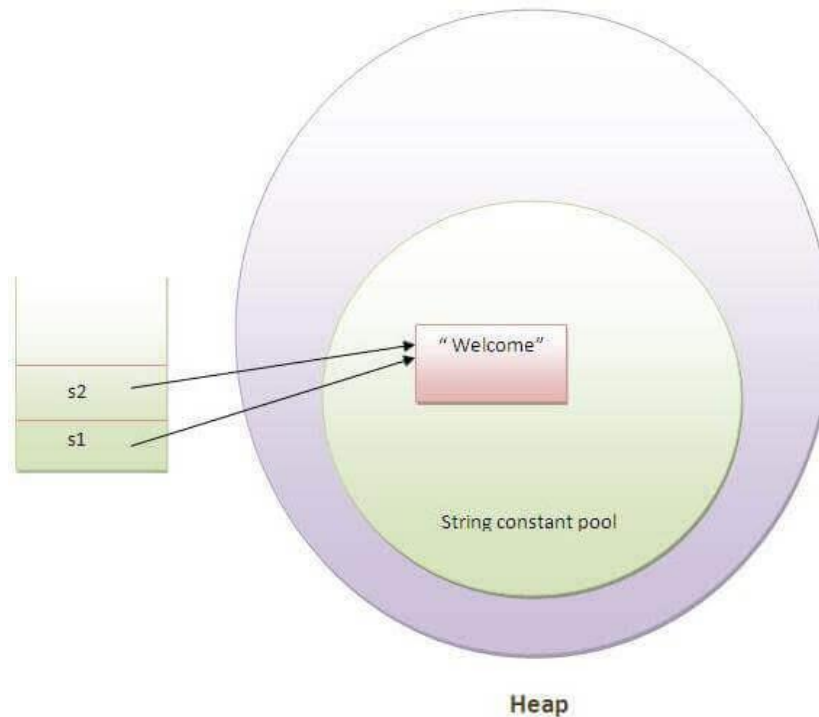
1) String Literal

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**".

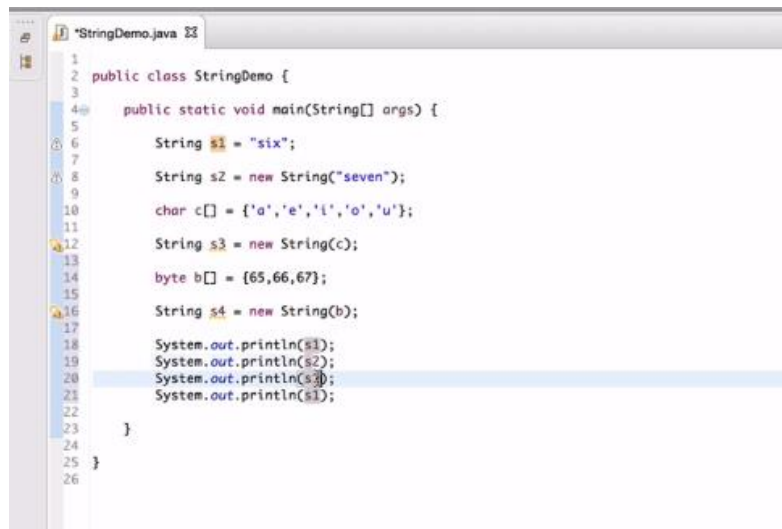
Why Java uses the concept of String literal?

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

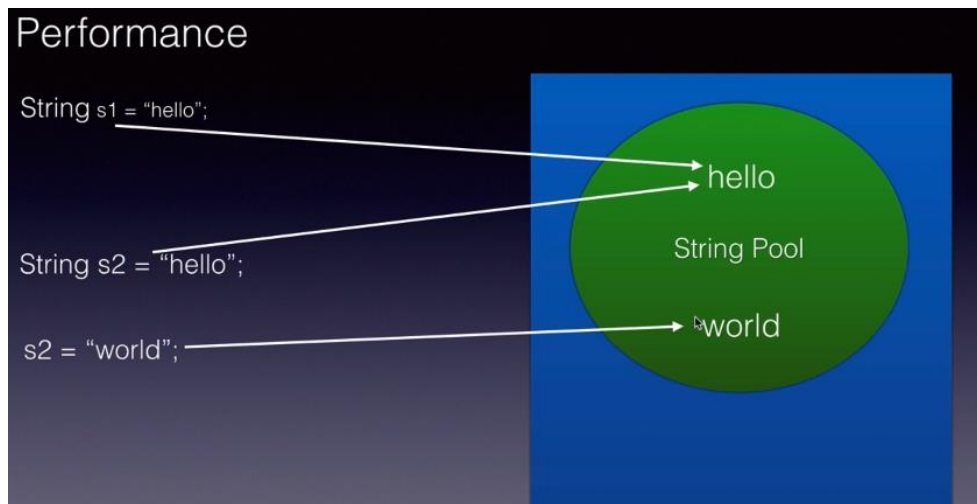
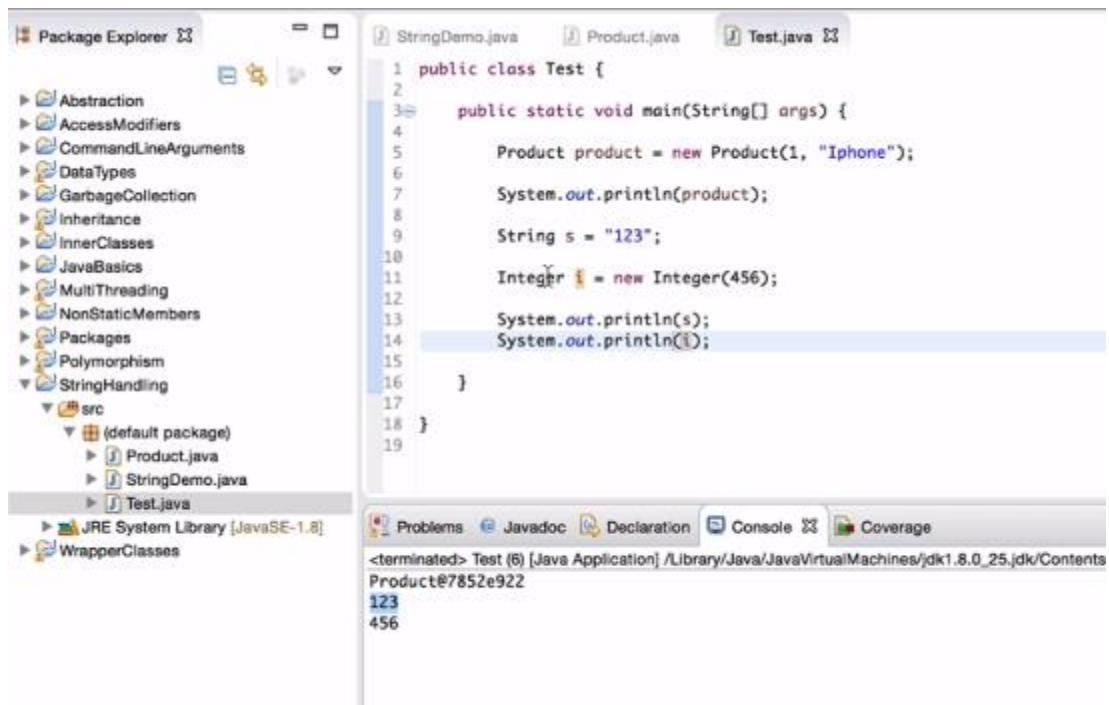
2) By **new keyword**

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).



```
1 public class StringDemo {
2
3
4     public static void main(String[] args) {
5
6         String s1 = "six";
7
8         String s2 = new String("seven");
9
10        char c[] = {'a','e','l','o','u'};
11
12        String s3 = new String(c);
13
14        byte b[] = {65,66,67};
15
16        String s4 = new String(b);
17
18        System.out.println(s1);
19        System.out.println(s2);
20        System.out.println(s3);
21        System.out.println(s4);
22
23    }
24
25 }
26
```



Example program to show both are of same locations

```
Product.java StringDemo.java Test.java User.java Test.java
1 package immutable;
2
3 public class StringPoolDemo {
4
5     public static void main(String[] args) {
6
7         User user1 = new User(1, "abc");
8         User user2 = new User(2, "abc");
9
10        System.out.println(user1);
11        System.out.println(user2);
12
13        String s1 = "abc";
14        String s2 = "abc";
15        String s3 = "abc";
16        System.out.println(s1.hashCode());
17        System.out.println(s2.hashCode());
18        System.out.println(s3.hashCode());
19    }
20 }
21
22 }
23
```

Problems Javadoc Declaration Console Coverage

<terminated> StringPoolDemo [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/bin/java -Djava.class.path=. StringPoolDemo
immutable.User@7852e922
immutable.User@4e25154f
96354
96354
96354

Concat Operation

```
Product.java StringDemo.java Test.java User.java Test.java
1 package immutable;
2
3 public class ImmutableDemo {
4
5     public static void main(String[] args) {
6
7         String s1 = "Hello";
8         String s2 = "World";
9
10        System.out.println("Before Concat:" + "\ns1:" + s1 + "\ns2:" + s2);
11
12        s1 = s1.concat(s2);
13
14        System.out.println("After Concat:" + "\ns1:" + s1 + "\ns2:" + s2);
15
16    }
17 }
18 }
19
```

Problems Javadoc Declaration Console Coverage

<terminated> ImmutableDemo [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/bin/java -Djava.class.path=. ImmutableDemo
Before Concat:
s1:Hello
s2:World
After Concat:
s1:HelloWorld
s2:World

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

```
1 package immutable;
2
3 public class StringPoolDemo {
4
5     public static void main(String[] args) {
6
7         User user1 = new User(1, "abc");
8         User user2 = new User(2, "abc");
9
10        System.out.println(user1);
11        System.out.println(user2);
12
13        String s1 = "abc";
14        String s2 = "xyz";
15        String s3 = "abc";
16        String s4 = new String("abc");
17        System.out.println(s1.hashCode());
18        System.out.println(s2.hashCode());
19        System.out.println(s3.hashCode());
20
21        System.out.println(s1 == s3);
22        System.out.println(s1.equals(s3));
23
24        System.out.println(s1 == s2);
25        System.out.println(s1.equals(s2));
26
27        System.out.println(s1 == s4);
28        System.out.println(s1.equals(s4));
29
30    }
31 }
```

Utility Methods in String class ...

a)indexOf

b)charAt

```
1 public class StringMethods {
2
3     public static void main(String[] args) {
4         String s = "Hello";
5
6         System.out.println("Length of the string is:" + s.length());
7         System.out.println("indexOf:" + s.indexOf('o'));
8         System.out.println("charAt:" + s.charAt(3));
9
10        System.out.println("substring with beginning index:" + s.substring(2));
11        System.out.println("substring with begin and end index:" + s.substring(0, 4));
12
13    }
14 }
15 }
```

c)substring() {overloaded methods}

d)replace()

e)toUpperCase()

f)toLowerCase()

g)split();

h)replace();

i>equals

j)equalsIgnoreCase()

```
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)

```
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, 5));  
    }  
}
```

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

String Builder vs String Buffer ?

StringBuffer vs 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

Java Enums

An **enum** is a special "class" that represents a group of **constants** (unchangeable variables, like **final** variables).

To create an **enum**, use the **enum** keyword (instead of class or interface), and separate the constants with a comma. Note that they should be in uppercase letters:

This concept is introduced in 1.5 version.

by default enum constants are **public static final**.

Example :

```
public enum Grades {  
    HIGH,  
    LOW,  
    MEDIUM  
}
```

- Inside the enum it is possible to declare constructors. That constructors will be executed for each and every constant. If we are declaring 5 constants then 5 times constructor will be executed.

- Inside the enum if we are declaring only constants the semicolon is optional.
- Inside the enum if we are declaring group of constants and constructors at that situation the group of constants must be first line of the enum must ends with semicolon.