String is basically an object that represents sequence of char values. String in Java is very special class provides to create and manipulate strings.

**String are Immutable in Java**

String objects are immutable in java, means it is unmodifiable. Once string object created its data or state can't be changed but a new String object created. Now it can be understood by diagram given below. Here coding is not changed but a new object is created with codingninja, thats why string is known as immutable.

Because String is immutable, it is not efficient to use String if you need to modify your string frequently (that would create many new Strings occupying new storage areas). For example,

// inefficient codes

String str = "Hello";

for (int i = 1; i < 1000; ++i) {

str = str + i;

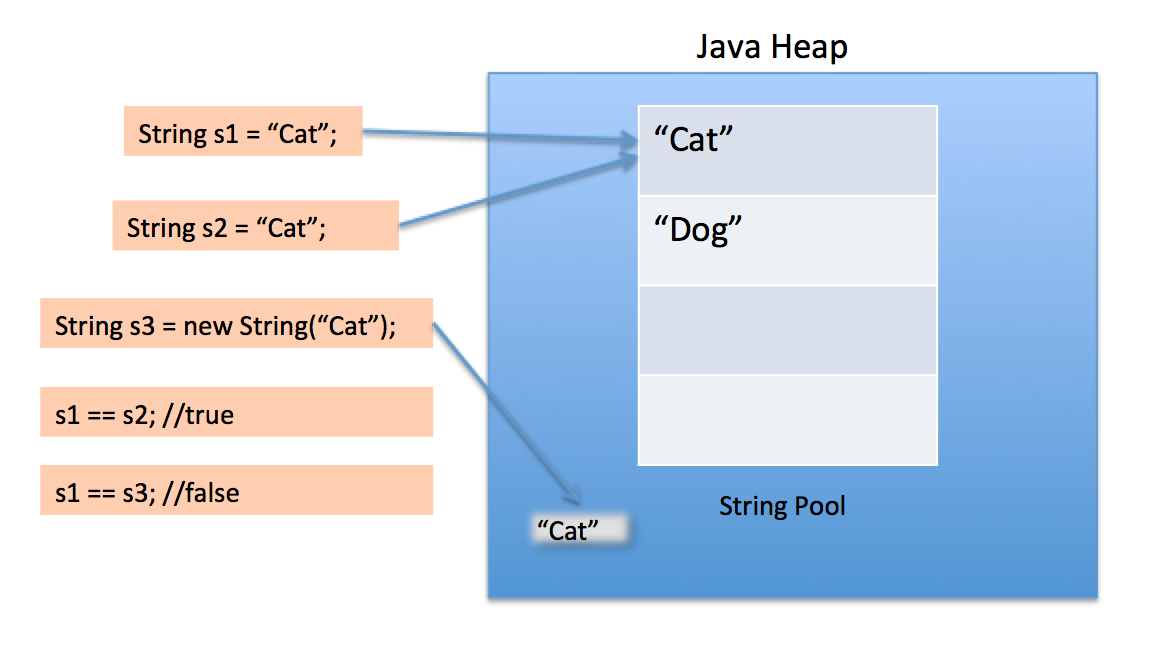
}

**Why String objects are immutable in java ?**

Suppose there are 100 reference variables all refers to one object "coding". If one reference variable changes the value of the object, it will be affected to all the reference variables. That is why string objects are immutable in java

**String Pool**

As the name suggests, String Pool in java is a pool of Strings stored in Java Heap Memory. We know that String is special class in java and we can create String object using new operator as well as providing values in double quotes.

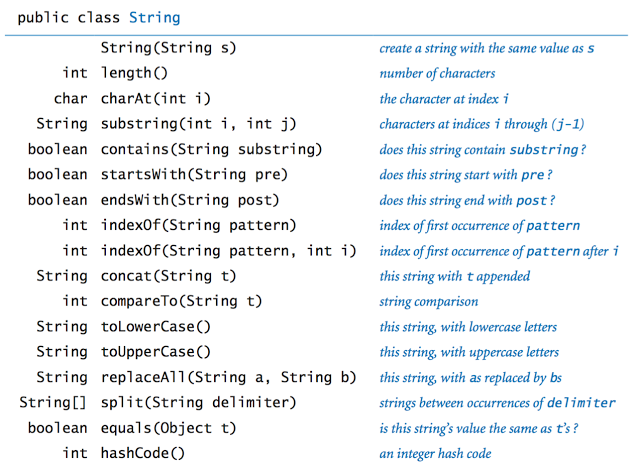


**Heap Space**

When a Java program started Java Virtual Machine gets some memory from Operating System. Java Virtual Machine or JVM uses this memory for all its need and part of this memory is call java heap memory. Heap in Java generally located at bottom of address space and move upwards. whenever we create object using new operator or by any another means object is allocated memory from Heap and When object dies or garbage collected ,memory goes back to Heap space in Java

**String Methods:**

Here is the list of methods supported by String class:



**Strings length:**

It returns the number of characters contained in the string.

public class StringDemo {

public static void main(String args[]) {

String palindrome = "We study in codingninja";

int len = palindrome.length();

System.out.println( "String Length is : " + len );

}

}

Output:

String Length is : 23

**Note:**

Strings in java are not terminated by ‘\0’ (null character) unlike c/c++.

**char At:**

This method returns the character located at the String's specified index. The string indexes start from zero. Here is the syntax of this method: public char charAt(int index)

Example:

public class Test {

public static void main(String args[]) {

String s = "Strings are immutable";

char result = s.charAt(8);

System.out.println(result);

}

}

Result:

a

**Concatenating Strings:**

The String class includes a method for concatenating two strings: string1.concat(string2); This returns a new string that is string1 with string2 added to it at the end. You can also use the concat() method with string, as in:

"My name is ".concat("Zara");

Strings are more commonly concatenated with the + operator, as in:

"Hello," + " world" + "!"

output: "Hello, world!"

Let us look at the following example:

public class StringDemo {

public static void main(String args[]) {

String string1 = "We are coding ";

System.out.println("Hey " + string1 + "Ninjas");

}

}

Result:

Hey we are coding Ninjas

**Strings Substring:**

This method has two variants and returns a new string that is a substring of this string.

Syntax:

public String substring(int beginIndex)

or

public String substring(int beginIndex, int endIndex) //beginIndex -- inclusive & endIndex -- exclusive.

Example:

public class Test{

public static void main(String args[]){

String Str = new String("Welcome to codingninjas.in");

System.out.print("Return Value :" );

System.out.println(Str.substring(10) );

System.out.print("Return Value :" );

System.out.println(Str.substring(10, 15) );

}

}

Result:

Return Value : codingninjas.in

Return Value : codi

**String Conversion**

Different ways to convert primitive types to String :

String str = String.valueOf(int\_number)

String str = Integer.toString(int\_number)

String str = “”+int\_number

From String to integer :

int num = Integer.parseInt(int\_num\_string)

Similarly can be done for other primitive types

**"==" vs .equals():**

Use the string.equals(String other) function to compare strings, not the == operator. The equal() method checks the actual contents of the string, the == operator checks whether the addresses to the Strings are equal.

public static void main(String[] args){

String str1 = "hello";

String str2 = "hello";

if(str2 == str1){

System.out.println("a) == match");

}

if(str2.equals(str1)){

System.out.println("b) equals match");

}

String str3 = new String("hello");

String str4 = new String("hello");

if(str4 == str3){

System.out.println("c) == match");

}

if(str2.equals(str3)){

System.out.println("d) equals match");

}

if(str4.equals(str3)){

System.out.println("e) equals match");

}

}

Output:

a) == match

b) equals match

d) equals match

e) equals match

Hence we should always use equals function to compare strings.

**String array**

Square brackets is used to declare a String array. There are two ways of using it. The first one is to put square brackets after the String reserved word. For example:

//Declaration

String thisIsAStringArray[];

//or

String[] thisIsAStringArray;

Both statements will declare the variable "thisIsAStringArray" to be a String Array. Note that this is just a declaration, the variable "thisIsAStringArray" will have the value null. And since there is only one square brackets, this means that the variable is only a one-dimensional String Array.

//Initialization

String[] thisIsAStringArray = new String[5];

thisIsAStringArray[0] = "AAA";

thisIsAStringArray[1] = "BBB";

thisIsAStringArray[2] = "CCC";

thisIsAStringArray[3] = "DDD";

thisIsAStringArray[4] = "EEE";

or

String[] thisIsAStringArray = {"AAA", "BBB", "CCC", "DDD", "EEE"};

**Iterate Through String Array**

Since the size and contents of a String Array can vary, it is useful to iterate through all the values.

String[] thisIsAStringArray = {"Apple", "Banana", "Orange"};

for( int i = 0; i < thisIsAStringArray.length; i++)

{

String element = thisIsAStringArray[i];

System.out.println( element );

}

Output:

Apple

Banana

Orange