String str =”abc”;

is equivalent to:

char data[] = {‘a’, ‘b’, ‘c’};

String str = new String(data);

Let us now look at some of the inbuilt methods in String class.

Java String Methods

Java String length():

The Java String length() method tells the length of the string. It returns the count of the total number of characters present in the String. For example:

```
public class Example{
public static void main(String args[]{
String s1="hello";
String s2="whatsup";
System.out.println("string length is: "+s1.length());
```

```
System.out.println("string length is: "+s2.length());
}}
```

Here, String length() function will return the length 5 for s1 and 7 for s2 respectively.

Java String compareTo():

The Java String compareTo() method compares the given string with current string. It is a method of *'Comparable'* interface which is implemented by String class. Don't worry, we will be learning about String interfaces later. It either returns a positive number, negative number or 0. For example:

```
public class CompareToExample{
public static void main(String args[]){
String s1="hello";
String s2="hello";
String s3="hemlo";
String s4="flag";
```

```
System.out.println(s1.compareTo(s2)); // 0 because
both are equal
System.out.println(s1.compareTo(s3)); //-1 because
"l" is only one time lower than "m"
System.out.println(s1.compareTo(s4)); // 2 because
"h" is 2 times greater than "f"
}}
```

This program shows the comparison between the various string. It is noticed that

if s1 > s2, it returns a positive number

if s1 < s2, it returns a negative number

if s1 == s2, it returns 0

Java String concat() :

The Java String concat() method combines a specific string at the end of another string and ultimately returns a combined string. It is like appending another string. For example:

```
public class ConcatExample{
public static void main(String args[]){
String s1="hello";
s1=s1.concat("how are you");
System.out.println(s1);
}}
```

The above code returns “**hellohow are you**”.

Java String IsEmpty() :

This method checks whether the String contains anything or not. If the java String is Empty, it returns true else false. For example:

```
public class IsEmptyExample{
public static void main(String args[]){
String s1="";
String s2="hello";
System.out.println(s1.isEmpty());      // true
```

```
System.out.println(s2.isEmpty());      // false
}}
```

Java String Trim() :

The Java string trim() method removes the leading and trailing spaces. It checks the Unicode value of space character (‘\u0020’) before and after the string. If it exists, then removes the spaces and return the omitted string. For example:

```
public class StringTrimExample{
public static void main(String args[]){
String s1="  hello   ";
System.out.println(s1+"how are you");       // without trim()
System.out.println(s1.trim()+"how are you"); // with trim()
}}
```

In the above code, the first print statement will print “hello how are you” while the second statement will print “**hellohow are you**” using the trim() function.

Java String toLowerCase() :

The java string toLowerCase() method converts all the characters of the String to lower case. For example:

```
public class StringLowerExample{
public static void main(String args[]){
String s1="HELLO HOW Are You?”;
String s1lower=s1.toLowerCase();
System.out.println(s1lower);}
}
```

The above code will return “**hello how are you**”.

Java String toUpper() :

The Java String toUpperCase() method converts all the characters of the String to upper case. For example:

```
public class StringUpperExample{
public static void main(String args[]){
String s1="hello how are you";
String s1upper=s1.toUpperCase();
System.out.println(s1upper);
}}
```

The above code will return “**HELLO HOW ARE YOU**”.

Java String ValueOf():

This method converts different types of values into a string. Using this method, you can convert int to string, long to string, Boolean to string, character to string, float to a string, double to a string, object to string and char array to string. The signature or syntax of string valueOf() method is given below: public static String valueOf(boolean b)

public static String valueOf(char c)

public static String valueOf(char[] c)

public static String valueOf(int i)

public static String valueOf(long l)

public static String valueOf(float f)

public static String valueOf(double d)

public static String valueOf(Object o)

Let's understand this with a programmatic example:

```
public class StringValueOfExample{
public static void main(String args[]){
int value=20;
String s1=String.valueOf(value);
System.out.println(s1+17);      //concatenating string
with 10
}}
```

In the above code, it concatenates the Java String and gives the output — **2017.**

Java String replace():

The Java String replace() method returns a string, replacing all the old characters or CharSequence to new characters. There are 2 ways to replace methods in a Java String.

```
public class ReplaceExample1{
public static void main(String args[]){
String s1="hello how are you";
String replaceString=s1.replace('h','t');
System.out.println(replaceString); }}
```

In the above code, it will replace all the occurrences of '**h**' to '**t**'. Output to the above code will be "**tello tow are you**". Let's see another type of using replace method in java string:

Java String replace(CharSequence target, CharSequence replacement) method :

```
public class ReplaceExample2{
public static void main(String args[]){
String s1="Hey, welcome to Edureka";
String
replaceString=s1.replace("Edureka","Brainforce");
System.out.println(replaceString);
}}
```

In the above code, it will replace all occurrences of "**Edureka**" to "**Brainforce**". Therefore, the output would be " **Hey, welcome to Brainforce**".

Java String contains() :

The Java string contains() method searches the sequence of characters in the string. If the sequences of characters are found, then it returns true otherwise returns false. For example:

```
class ContainsExample{
public static void main(String args[]){
String name=" hello how are you doing?";
System.out.println(name.contains("how are you"));  //
returns true
System.out.println(name.contains("hello"));       //
returns true
System.out.println(name.contains("fine"));        //
returns false
}}
```

In the above code, the first two statements will return true as it matches the String whereas the second print statement will return false because the characters are not present in the string.

Java String equals() :

The Java String equals() method compares the two given strings on the basis of the content of the string

i.e Java String representation. If all the characters are matched, it returns true else it will return false. For example:

```
public class EqualsExample{
public static void main(String args[]){
String s1="hello";
String s2="hello";
String s3="hi";
System.out.println(s1.equalsIgnoreCase(s2));   //
returns true
System.out.println(s1.equalsIgnoreCase(s3));   //
returns false
}
}
```

Java String equalsIgnoreCase():

This method compares two string on the basis of content but it does not check the case like equals()

method. In this method, if the characters match, it returns true else false. For example:

```
public class EqualsIgnoreCaseExample{
public static void main(String args[]){
String s1="hello";
String s2="HELLO";
String s3="hi";
System.out.println(s1.equalsIgnoreCase(s2));   //
returns true
System.out.println(s1.equalsIgnoreCase(s3));   //
returns false
}}
```

In the above code, the first statement will return true because the content is the same irrespective of the case. Then, in the second print statement will return false as the content doesn’t match in the respective strings.

Java String toCharArray():

This method converts the string into a character array i.e first it will calculate the length of the given Java String including spaces and then create an array of char type with the same content. For example:

```
StringToCharArrayExample{
public static void main(String args[]){
String s1="Welcome to Edureka";
char[] ch=s1.toCharArray();
for(int i=0;i<ch.length;i++){
System.out.print(ch[i]);
}}}
```

The above code will return "**Welcome to Edureka**".

Java StringGetBytes() :

The Java string getBytes() method returns the sequence of bytes or you can say the byte array of the string. For example:

```
public class StringGetBytesExample {
public static void main(String args[]){
String s1="ABC";
byte[] b=s1.getBytes();
for(int i=0;i<b.length;i++){
System.out.println(b[i]);
}
}}
```

In the above code, it will return the value **65,66,67.**

Java String IsEmpty() :

This method checks whether the String is empty or not. If the length of the String is 0, it returns true else false. For example:

```
public class IsEmptyExample{
public static void main(String args[]) {
String s1="";
String s2="hello";
System.out.prinltn(s1.isEmpty());     // returns true
System.out.prinltn(s2.isEmpty());     // returns false
}}
```

In the above code, the first print statement will **return true** as it does not contain anything while the second print statement will **return false**.

Java String endsWith() :

The Java String endsWith() method checks if this string ends with the given suffix. If it returns with the given suffix, it will return true else returns false. For example:

```
public class EndsWithExample{
public static void main(String args[]) {
```

```
String s1="hello how are you”;
System.out.println(s1.endsWith("u"));       // returns
true
System.out.println(s1.endsWith("you"));     // returns
true
System.out.println(s1.endsWith("how"));     // returns
false
}}
```

This is not the end. There are more Java String methods that will help you make your code simpler.

Moving on, Java String class implements three **interfaces**, namely **— *Serializable, Comparable*** and ***CharSequence***.

String Handling

String is an array of characters in many programming languages.In Java String is also an array of characters.

But mainly a String is a class in Java.It is a predefined class in

java.lang package.

Java.lang package is a default package in Java.

There are two String Handling classes available in Java namely

String

StringBuffer

Both String and StringBuffer are final classes.

Both these classes are available in java.lang package.

String :

It is a final class in java.lang package.

A String class is an object in Java.

A String is immutable, which means that it is fixed and the

contents of a String are always constant.

String constructors :

We can create an object of a String class by different ways by

using different types of constructors.

String s1=new String()

String s2=new String(“program”);

String s3=new String(s2);

String s4=”program”;

char name[]={‘J’,’I’,’T’,’M’};

String s5=new String(name);

There are many methods available in String class.

Some of the important methods are:

length():

This method is used to find the length or size of a String.This

method returns number of characters present in the given String.

Syntax of the method:

int length()

Eg:

String s=”CoreJava”;

int i=s.length();

length() returns the length of the string and its length is stored in

i which is 8

concat():

This method is used to join or concat two strings.

The return type of this method is String.

This method takes one parameter which is also a String

String concat(String str)

Where str is the String joined to the given String.

Eg:

String s1=”Java”;

String s2=”Program”;

String s3=s1.concat(s2);

Where the output of s3 is JavaProgram

Concatenation operator(+):

The operator “+” is called a concatenation operator.

It is similar to concat() method.

In Java + is used to join any strings.

Eg:

String s1=”Java”;

String s2=”Program”;

String s3=s1+s2;

Where the output of s3 is JavaProgram

toLowerCase():

This method is used to convert upper case letters of a String to

lowercase.The return type of this method is String.

String toLowerCase()

Eg:

String s=”JAVA PROGRAM”;

The output of

s.toLowerCase() is java program

toUpperCase():

This method is used to convert lower case letters of a String to

uppercase.The return type of this method is String.

String toUpperCase()

Eg:

String s=" java";

The output of

s.toLowerCase() is JAVA

trim():

This method is used to filter white spaces before and after a

given String.

This mehod is not used to filter white spaces between any two

words of a String.

The return type of this method is String

String trim()

String s=” Learn Java Basics “;

The output of

s.trim() is Learn Java Basics

replace():

This method is used to replace a given character of a String with

another new character . The return type of this method is String.

This method replaces the occurrence of the given character

through out the String by the new character.

String replace(char original,char new)

Eg:

String s=”Liver”;

The output of

s.replace(‘L’,’R’) is River

String Comparision Methods:

There are three types of String Comparision in Java.

They are :

(i)equals()

(ii)==

(iii)compareTo()

equals():

This method is used to compare actual contents of the two

strings.The return type of this method is a boolean value.

boolean equals(String s)

Eg:

String s1=new String(“rama”);

String s2=new String(“jaya”);

String s3=new String(“rama”);

String s4=new String(“RAMA”);

The output of

if(s1.equals(s2)) is false

if(s1.equals(s4)) is false

if(s1.equals(s3)) is true

equalsIgnoreCase():

This method is used to compare actual contents of the two

strings same as equals() except that it neglects case

sensitiveness of letters.

Eg:

String s1=new String(“rama”);

String s2=new String(“RAMA”);

The output of

if(s1.equals(s2)) is false

if(s1.equalsIgnoreCase(s3)) is true

compareTo():

This method is used to compare two strings depending on the

ASCII values of the characters of the string.

The return type of this method is int.

int compareTo(String s)

Eg:

String s1=”A123”;

String s2=”C123”;

String s3=”A123”;

It returns,

<0 if s1 is less than s2 and the output of

S1.compareTo(s2) is -2

>0 if s1 is greater than s2 and the output of

s2.compareTo(s1) is 2

=0 if s1 is equal to s2 and he output of

s1.compareTo(s3) is 0

Character Extraction:

The String class provides a number of ways in which

characters can be extracted from a String object.There are many

methods used to extract characters from a String.The important

method is charAt()

charAt():

This method is used to extract a single character from a

String.The return type of this method is char.

char charAt(int where)

Here, where is the index of the character.

Eg;

String s=”ABCDEF”;

s.charAt(2);

where the output is C,since at location 2 there is character C.

Modifying a String:

There are many ways of modifying a String.Mostly used

modifying method is substring().

substring():

This method is used to return a substring of the given String

starting from the startindex of the given method.

The return type of this method is also a String.

This method is overloaded.

(i)String substring(int startindex)

It returns a substring starting from startindex to the end of the

String.

Eg:

String s=”Learn from the Basic Concepts”;

012345---posions of characters in the String

The output of

s.substring(5) is from the Basic Concepts

(ii)String substring(int startindex,int endindex):

It returns a substring starting from startindex upto the end of the

end index without including endindex.

Eg:

String s=”Learn from the Basic Concepts”;

0123456789

s.substring(5,9) is from

In this character in the location of 9 will not be included.

StringBuffer:It is a final class in java.lang package.

A StringBuffer is mutable,which means that which means that it

is not fixed and the contents of a StringBuffer may change.It is

dynamic in nature,where we can insert any character at any

location.

A StringBuffer class stores an additional memory for 16

characters.

There are some methods which are available only in StringBuffer

class only.Some of the important methods are:

capacity():

This method is used to return than memory capacity of a StringBuffer.The return type of this method is int.

Eg:

StringBuffer sb1=new StringBuffer();

StringBuffer sb2=new StringBuffer("ABCDE");

Sb1.length() is 0 and sb2.length() is 5

sb1.capacity() is 16 where as

sb2.capacity is 21

append():

This method is similar to concat() method in String class.This method is used to join any string to the StringBuffer.

The return type of this method is StringBuffer.

StringBuffer append(String s)

Eg:

StringBuffer sb=new StringBuffer("Learn");

Sb.append("Basics");

Where the output is LearnBasics

reverse():

This method is used to reverse the characters of a

given StringBuffer.The return type of this method is again a

StringBuffer.

StringBuffer reverse()

Eg:

StringBuffer sb=new StringBuffer(“ABCDEF”);

sb.reverse();

Where the output is FEDCBA