**5. String handling**

Generally, string is a sequence of characters. But in java, string is an object that represents a sequence of characters. The java.lang.String class is used to create string object.  
 For example:

1. char[] ch={'j','a','v','a'};
2. String s=new String(ch);

is same as:

1. String s="java";

### How to create String object?

|  |
| --- |
| There are two ways to create String object:   1. By string literal 2. By new keyword |

### String Literal

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

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 string doesn't exist in the pool, a new string instance is created and placed in the pool. For example:

String s1="Welcome";    
String s2="Welcome";//will not create new instance

### By new keyword String s=new String("Welcome");//creates two objects and one reference variable

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.

### Example

public class StringExample{   output:

public static void main(String args[]){   java

String s1="java";//creating string by java string literal   strings

char ch[]={'s','t','r','i','n','g','s'};   example

String s2=new String(ch);//converting char array to string

String s3=new String("example");//creating java string by new keyword

System.out.println(s1);

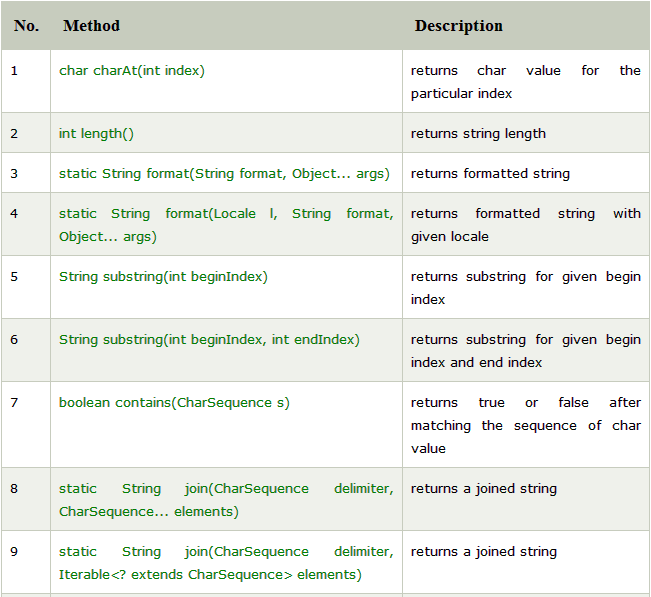
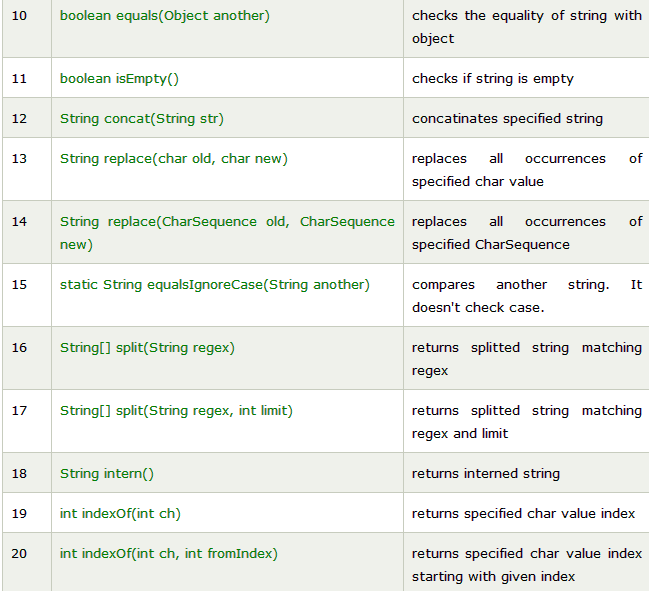
System.out.println(s2);

System.out.println(s3);

}}

### Java String class methods

The java.lang.String class provides many useful methods to perform operations on sequence of char values.



**String Concatenation in Java**

In java, string concatenation forms a new string *that is* the combination of multiple strings. There are two ways to concat string in java:

1. By + (string concatenation) operator
2. By concat() method.

## 1) String Concatenation by + (string concatenation) operator

Java string concatenation operator (+) is used to add strings. For Example:

class TestStringConcatenation1  
{   Output: Sachin Tendulkar

 public static void main(String args[])  
{

   String s="Sachin"+" Tendulkar";

   System.out.println(s);//

 }

}

### 2) String Concatenation by concat() method

The String concat() method concatenates the specified string to the end of current string. Syntax:

public String concat(String another)

Let's see the example of String concat() method.

class TestStringConcatenation3{

 public static void main(String args[]){

   String s1="Sachin ";

   String s2="Tendulkar";

   String s3=s1.concat(s2);

   System.out.println(s3);//Sachin Tendulkar

  }

}

**Java String compareTo()**

The **java string compareTo()** method compares the given string with current string lexicographically. It returns positive number, negative number or 0.

It compares strings on the basis of Unicode value of each character in the strings.

If first string is lexicographically greater than second string, it returns positive number (difference of character value). If first string is less than second string lexicographically, it returns negative number and if first string is lexicographically equal to second string, it returns 0.

1. if s1 > s2, it returns positive number
2. if s1 < s2, it returns negative number
3. if s1 == s2, it returns 0

Ex: public class CompareToExample{

public static void main(String args[]){

String s1="hello";

String s2="hello";

String s3="meklo";

String s4="hemlo";   output:

System.out.println(s1.compareTo(s2)); 0

System.out.println(s1.compareTo(s3)); -5

System.out.println(s1.compareTo(s4)); -1

}}

**Java String length**

The **java string length()** method length of the string. It returns count of total number of characters. The length of java string is same as the unicode code units of the string.

public int length()

public class LengthExample{

public static void main(String args[]){

String s1="java";

String s2="python";

System.out.println("string length is: "+s1.length());

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

}}

**String searching**

# Java String indexOf()

The **java string indexOf()** method returns index of given character value or substring. If it is not found, it returns -1. The index counter starts from zero.

There are 4 types of indexOf method in java. The signature of indexOf methods are given below:

|  |  |  |
| --- | --- | --- |
| **No.** | **Method** | **Description** |
| 1 | int indexOf(int ch) | returns index position for the given char value |
| 2 | int indexOf(int ch, int fromIndex) | returns index position for the given char value and from index |
| 3 | int indexOf(String substring) | returns index position for the given substring |
| 4 | int indexOf(String substring, int fromIndex) | returns index position for the given substring and from index |

Ex: public class IndexOfExample{

public static void main(String args[]){

String s1="this is index of example";

int index1=s1.indexOf("is"); //returns the index of is substring

int index2=s1.indexOf("index"); //returns the index of index substring

System.out.println(index1+"  "+index2); //2 8

int index3=s1.indexOf("is",4); //returns the index of is substring after 4th index

System.out.println(index3); //5 i.e. the index of another is

int index4=s1.indexOf('s'); //returns the index of s char value

System.out.println(index4); //3

}}

**Java String replace**

The **java string replace()** method returns a string replacing all the old char or CharSequence to new char or CharSequence.

There are two type of replace methods in java string.

public String replace(char oldChar, char newChar)  and

public String replace(CharSequence target, CharSequence replacement)

public class ReplaceExample1{

public static void main(String args[]){

String s1="javatpoint is a very good website";

String replaceString=s1.replace('a','e');

System.out.println(replaceString);

}}

Output: jevetpoint is e very good website

**Character Extraction in Java**

There are several ways by which characters can be extracted from String class object. String is treated as an object in Java so we can’t directly access the characters that comprise a string. For doing this String class provides various predefined methods.

### charAt()

charAt() method is used to extract a single character at an index. It has following syntax.

Syntax: char charAt(int index)

### getChars()

It is used to extract more than one character. getChars() has following syntax.

void getChars(int stringStart, int stringEnd, char arr[], int arrStart)

### getBytes()

getBytes() extract characters from String object and then convert the characters in a byte array. It has following syntax.

byte [] getBytes()

### toCharArray()

It is an alternative of getChars() method. toCharArray() convert all the characters in a String object into an array of characters. It is the best and easiest way to convert string to character array. It has following syntax

char [] toCharArray()

**Java Convert String to int**

We can convert **String to int in java** using Integer.parseInt() method. To convert String into Integer, we can use Integer.valueOf() method which returns instance of Integer class.

The parseInt() is the static method of Integer class. The **signature** of parseInt() method is given below:

public static int parseInt(String s)

public class StringToIntExample{

public static void main(String args[]){

String s="200";

int i=Integer.parseInt(s);

System.out.println(s+100);

System.out.println(i+100);

}}

Output:

200100

300

**Java String toLowerCase()**

The **java string toLowerCase()** method returns the string in lowercase letter. In other words, it converts all characters of the string into lower case letter.

public String toLowerCase()

public String toLowerCase(Locale locale)

public class StringLowerExample{

public static void main(String args[]){

String s1="JAVATPOINT HELLO stRIng";

String s1lower=s1.toLowerCase();

System.out.println(s1lower);

}}

Output: javatpoint hello string

**Java String toUpperCase**

The **java string toUpperCase()** method returns the string in uppercase letter. In other words, it converts all characters of the string into upper case letter.

There are two variant of toUpperCase() method. The signature or syntax of string toUpperCase() method is given below:

public String toUpperCase()

public String toUpperCase(Locale locale)

public class StringUpperExample{

public static void main(String args[]){

String s1="hello string";

String s1upper=s1.toUpperCase();

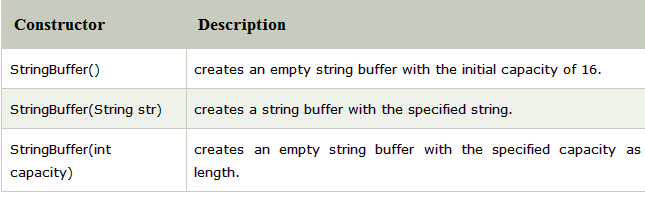
System.out.println(s1upper);

}}

Output: HELLO STRING

**java StringBuffer class**

Java StringBuffer class is used to create mutable (modifiable) string. The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.





### StringBuffer delete() method

The delete() method of StringBuffer class deletes the string from the specified beginIndex to endIndex.

class StringBufferExample4{

public static void main(String args[]){

StringBuffer sb=new StringBuffer("Hello");

sb.delete(1,3);

System.out.println(sb);//prints Hlo

}

}

### StringBuffer replace() method

The replace() method replaces the given string from the specified beginIndex and endIndex.

class StringBufferExample3{

public static void main(String args[]){

StringBuffer sb=new StringBuffer("Hello");

sb.replace(1,3,"Java");

System.out.println(sb);//prints HJavalo

}

}