For example:

Testimmutablestring1.java

```
class Testimmutablestring1{
public static void main(String args[]){
String s="Sachin";
s=s.concat(" Tendulkar");
System.out.println(s);
}
}
```

Output:

```
Sachin Tendulkar
```

In such a case, s points to the "Sachin Tendulkar". Please notice that still Sachin object is not modified.

String methods

Java String charAt()

The **Java String class charAt()** method returns *a char value at the given index number*.

The index number starts from 0 and goes to n-1, where n is the length of the string. It returns **StringIndexOutOfBoundsException,** if the given index number is greater than or equal to this string length or a negative number.

Syntax

```
public char charAt(int index)
public class CharAtExample{
public static void main(String args[]){
String name="javatsupper";
char ch=name.charAt(4);//returns the char value at the 4th index
System.out.println(ch);
}}
```

String compare

We can compare String in Java on the basis of content and reference.

It is used in **authentication** (by equals() method), **sorting** (by compareTo() method), **reference matching** (by == operator) etc.

There are three ways to compare String in Java:

1. By Using equals() Method
2. By Using == Operator
3. By compareTo() Method

equals() Method

The String class equals() method compares the original content of the string. It compares values of string for equality. String class provides the following two methods:

Using == operator

The == operator compares references not values.

compareTo() method

The String class compareTo() method compares values lexicographically and returns an integer value that describes if first string is less than, equal to or greater than second string.

Suppose s1 and s2 are two String objects. If:

- **s1 == s2** : The method returns 0.
- **s1 > s2** : The method returns a positive value.
- **s1 < s2** : The method returns a negative value.

String Concatenation in Java

String Concatenation by + (String concatenation) operator

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

TestStringConcatenation1.java

```
class TestStringConcatenation1{
 public static void main(String args[]){
   String s="Sachin"+" Tendulkar";
   System.out.println(s);//Sachin Tendulkar
 }
}
```

- ## Concatenation by concat() method

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

1. **public** String concat(String another)

Substring in Java

A part of String is called **substring**. In other words, substring is a subset of another String. Java String class provides the built-in substring() method that extract a substring from the given string by using the index values passed as an argument. In case of substring() method startIndex is inclusive and endIndex is exclusive.

Suppose the string is "**computer**", then the substring will be com, compu, ter, etc.

String endsWith()

The **Java String class endsWith()** method checks if this string ends with a given suffix. It returns true if this string ends with the given suffix; else returns false.

Signature

The syntax or signature of endsWith() method is given below.

```
1. public boolean endsWith(String suffix)
```

getBytes() Method Example

The parameterless getBytes() method encodes the string using the default charset of the platform, which is UTF - 8. The following two examples show the same.

FileName: StringGetBytesExample.java

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

Output:

```
65
66
67
68
69
70
71
```

String indexOf()

The **Java String class indexOf()** method returns the position of the first occurrence of the specified character or string in a specified string.

String isEmpty()

The **Java String class isEmpty()** method checks if the input string is empty or not. Note that here empty means the number of characters contained in a string is zero.

```
public class IsEmptyExample{
public static void main(String args[]){
String s1="";
String s2="shuvra";

System.out.println(s1.isEmpty());
System.out.println(s2.isEmpty());
}}
```

Output:

```
true
false
```

Java StringBuffer Class

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

Important methods of StringBuffer class

Modifier and Type	Method	Description
public synchronized StringBuffer	append(String s)	It is used to append the specified string with this string. The append() method is overloaded like append(char), append(boolean), append(int), append(float), append(double) etc.
public synchronized StringBuffer	insert(int offset, String s)	It is used to insert the specified string with this string at the specified position. The insert() method is overloaded like insert(int, char), insert(int, boolean), insert(int, int), insert(int, float), insert(int, double) etc.
public synchronized StringBuffer	replace(int startIndex, int endIndex, String str)	It is used to replace the string from specified startIndex and endIndex.
public synchronized StringBuffer	delete(int startIndex, int endIndex)	It is used to delete the string from specified startIndex and endIndex.
public synchronized StringBuffer	reverse()	is used to reverse the string.
public int	capacity()	It is used to return the current capacity.
public void	ensureCapacity(int minimumCapacity)	It is used to ensure the capacity at least equal to the given minimum.
public char	charAt(int index)	It is used to return the character at the specified position.

public int	length()	It is used to return the length of the string i.e. total number of characters.
public String	substring(int beginIndex)	It is used to return the substring from the specified beginIndex.
public String	substring(int beginIndex, int endIndex)	It is used to return the substring from the specified beginIndex and endIndex.

What is a mutable String?

A String that can be modified or changed is known as mutable String. StringBuffer and StringBuilder classes are used for creating mutable strings.

Java StringBuilder Class

Java StringBuilder class is used to create mutable (modifiable) String. The Java StringBuilder class is same as StringBuffer class except that it is non-synchronized. It is available since JDK 1.5.

Important methods of StringBuilder class

Method	Description
public StringBuilder append(String s)	It is used to append the specified string with this string. The append() method is overloaded like append(char), append(boolean), append(int), append(float), append(double) etc.
public StringBuilder insert(int offset, String s)	It is used to insert the specified string with this string at the specified position. The insert() method is overloaded like insert(int, char), insert(int, boolean), insert(int, int), insert(int, float), insert(int, double) etc.
public StringBuilder replace(int startIndex, int endIndex, String str)	It is used to replace the string from specified startIndex and endIndex.
public StringBuilder delete(int startIndex, int endIndex)	It is used to delete the string from specified startIndex and endIndex.
public StringBuilder reverse()	It is used to reverse the string.

public int capacity()	It is used to return the current capacity.
public void ensureCapacity(int minimumCapacity)	It is used to ensure the capacity at least equal to the given minimum.
public char charAt(int index)	It is used to return the character at the specified position.
public int length()	It is used to return the length of the string i.e. total number of characters.
public String substring(int beginIndex)	It is used to return the substring from the specified beginIndex.
public String substring(int beginIndex, int endIndex)	It is used to return the substring from the specified beginIndex and endIndex.