**Today’s Topics**

**Java String**

1. Different Ways to Create String
2. Using string literal
3. Using new keyword
4. Java String compare
5. Java String Methods
   * split()
   * contains(CharSequence s)
   * length()
   * replace()
   * format()
   * substring()
6. String Concatenation
7. Java String Pool
8. String intern() Method
9. String to int conversion
10. String to float conversion
11. String to date conversion

**Java String**

* Basically, string is a sequence of characters but it’s not a primitive type.
* When we create a string in java, it actually creates an object of type String.
* String is immutable object which means that it cannot be changed once it is created.
* String is the only class where operator overloading is supported in java. We can concat two strings using + operator. For example "a"+"b"="ab".
* Java provides two useful classes for String manipulation – StringBuffer and StringBuilder.
* Important FAQ: Difference between String, StringBuffer and StringBuilder.

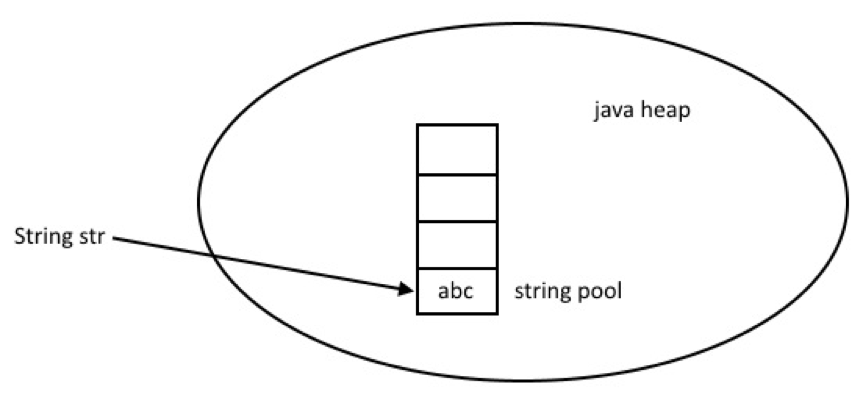
**Different Ways to Create String**

There are many ways to create string object in java, some of the popular ones are given below.

1. **Using string literal**

This is the most common way of creating string. In this case a string literal is enclosed with double quotes.

String str = "abc";

  
When we create a String using double quotes, JVM looks in the [String pool](https://www.journaldev.com/797/what-is-java-string-pool) to find if any other String is stored with same value. If found, it just returns the reference to that String object else it creates a new String object with given value and stores it in the String pool.

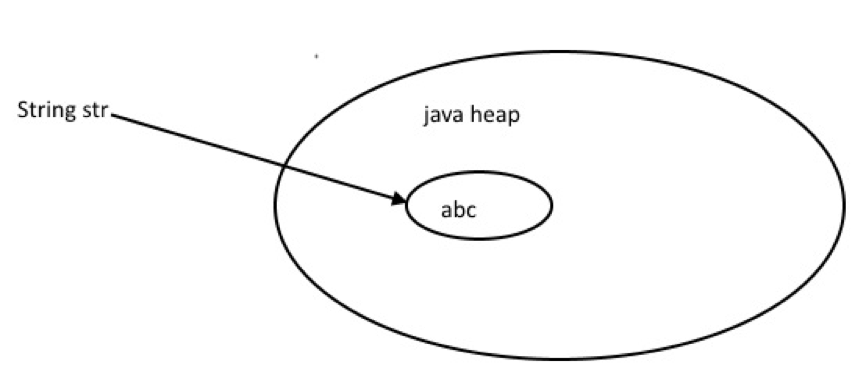
1. **Using new keyword**

We can create String object using new operator, just like any normal java class. There are several constructors available in String class to get String from char array, byte array, StringBuffer and StringBuilder.

String str = new String("abc");

char[] a = {'a', 'b', 'c'};

String str2 = new String(a);



**Java String compare**

String class provides equals() and equalsIgnoreCase() methods to compare two strings. These methods compare the value of string to check if two strings are equal or not. It returns true if two strings are equal and false if not.

package com.synergetics.string-demo.string.examples;

/\*\*

\* Java String Example

\*

\* @author Smita

\*

\*/

public class StringEqualExample {

public static void main(String[] args) {

//creating two string object

String s1 = "abc";

String s2 = "abc";

String s3 = "def";

String s4 = "ABC";

System.out.println(s1.equals(s2));//true

System.out.println(s2.equals(s3));//false

System.out.println(s1.equals(s4));//false;

System.out.println(s1.equalsIgnoreCase(s4));//true

}

}

Output of above program is:

true

false

false

true

String class implements [Comparable](https://www.journaldev.com/780/comparable-and-comparator-in-java-example) interface, which provides compareTo() and compareToIgnoreCase()methods and it compares two strings lexicographically.

Both strings are converted into Unicode value for comparison and returns integer value which can be greater than, less than or equal to zero. If strings are equal then it returns zero or else it returns either greater or less than zero.

package com.synergetics.string-demo.examples;

/\*\*

\* Java String compareTo Example

\*

\* @author Smita

\*

\*/

public class StringCompareToExample {

public static void main(String[] args) {

String a1 = "abc";

String a2 = "abc";

String a3 = "def";

String a4 = "ABC";

System.out.println(a1.compareTo(a2));//0

System.out.println(a2.compareTo(a3));//less than 0

System.out.println(a1.compareTo(a4));//greater than 0

System.out.println(a1.compareToIgnoreCase(a4));//0

}

}

Output of above program is:

0

-3

32

0

Please read this in more detail at [String compareTo example](https://www.journaldev.com/810/java-string-compareto-examples).

1. **Java String Methods**

Let’s have a look at some of the popular String class methods with example program.

1. split()

[Java String split](https://www.journaldev.com/791/java-string-split)() method is used to split the string using given expression. There are two variants of split() method.

* + split(String regex): This method splits the string using given regex expression and returns array of string.
  + split(String regex, int limit): This method splits the string using given regex expression and return array of string but the element of array is limited by the specified limit. If the specified limit is 2 then the method return an array of size 2.

package com.synergetics.string-demo.examples;

/\*\*

\* Java String split example

\*

\* @author Smita

\*

\*/

public class StringSplitExample {

public static void main(String[] args) {

String s = "a/b/c/d";

String[] a1 = s.split("/");

System.out.println("split string using only regex:");

for (String string : a1) {

System.out.println(string);

}

System.out.println("split string using regex with limit:");

String[] a2 = s.split("/", 2);

for (String string : a2) {

System.out.println(string);

}

}

}

**Output of above program is:**

*split string using only regex:*

*a*

*b*

*c*

*d*

*split string using regex with limit:*

*a*

*b/c/d*

1. **contains(CharSequence s)**

Java String contains() methods checks if string contains specified sequence of character or not. This method returns true if string contains specified sequence of character, else returns false.

package com.synergetics.string-demo.examples;

/\*\*

\* Java String contains() Example

\*

\* @author Smita

\*

\*/

public class StringContainsExample {

public static void main(String[] args) {

String s = "Hello World";

System.out.println(s.contains("W"));//true

System.out.println(s.contains("X"));//false

}

}

**Output of above program is:**

true

false

1. **length()**

[Java String length](https://www.journaldev.com/17817/java-string-length)() method returns the length of string.

package com.synergetics.string-demo.examples;

/\*\*

\* Java String length

\*

\* @author Smita

\*

\*/

public class StringLengthExample {

public static void main(String[] args) {

String s1 = "abc";

String s2 = "abcdef";

String s3 = "abcdefghi";

System.out.println(s1.length());//3

System.out.println(s2.length());//6

System.out.println(s3.length());//9

}

}

1. **replace()**

Java String **replace()** method is used to replace a specific part of string with other string. There are four variants of replace() method.

* + **replace(char oldChar, char newChar):** This method replace all the occurrence of oldChar with newChar in string.
  + **replace(CharSequence target, CharSequence replacement):** This method replace each target literals with replacement literals in string.
  + **replaceAll(String regex, String replacement**): This method replace all the occurrence of substring matches with specified regex with specified replacement in string.
  + **replaceFirst(String regex, String replacement):** This method replace first occurrence of substring that matches with specified regex with specified replacement in string.

package com.synergetics.string-demo.examples;

/\*\*

\* Java String replace

\*

\* @author Smita

\*

\*/

public class StringReplaceExample {

public static void main(String[] args) {

//replace(char oldChar, char newChar)

String s = "Hello World";

s = s.replace('l', 'm');

System.out.println("After Replacing l with m :");

System.out.println(s);

//replaceAll(String regex, String replacement)

String s1 = "Hello synergetics.string-demo, Hello Smita";

s1 = s1.replaceAll("Hello", "Hi");

System.out.println("After Replacing :");

System.out.println(s1);

//replaceFirst(String regex, String replacement)

String s2 = "Hello guys, Hello world";

s2 = s2.replaceFirst("Hello", "Hi");

System.out.println("After Replacing :");

System.out.println(s2);

}

}

**Output of above program is:**

After Replacing l with m :

Hemmo Wormd

After Replacing :

Hi synergetics.string-demo, Hi Smita

After Replacing :

Hi guys, Hello world

1. **format()**

Java Sting format() method is used to format the string. There is two variants of java String format() method.

* + **format(Locale l, String format, Object… args):** This method formats the string using specified locale, string format and arguments.
  + **format(String format, Object… args):** This method formats the string using specified string format and arguments.

package com.synergetics.string-demo.examples;

import java.util.Locale;

/\*\*

\* Java String format

\*

\* @author Smita

\*

\*/

public class StringFormatExample {

public static void main(String[] args) {

String s = "synergetics.string-demo.com";

// %s is used to append the string

System.out.println(String.format("This is %s", s));

//using locale as Locale.US

System.out.println(String.format(Locale.US, "%f", 3.14));

}

}

**Output of above program is:**

This is synergetics.string-demo.com

3.140000

1. **substring()**

This method returns a part of the string based on specified indexes.

package com.synergetics.string-demo.examples;

/\*\*

\* Java String substring

\*

\*/

public class StringSubStringExample {

public static void main(String[] args) {

String s = "This is synergetics.string-demo.com";

s = s.substring(8,18);

System.out.println(s);

}

}

**String Concatenation**

String concatenation is very basic operation in java. String can be concatenated by using “+” operator or by using concat() method.

package com.synergetics.string-demo.examples;

/\*\*

\* Java String concatenation

\*

\* @author Smita

\*

\*/

public class StringConcatExample {

public static void main(String[] args) {

String s1 = "Hello";

String s2 = "World";

String s3 = s1 + s2;

//using + operator

System.out.println("Using + operator: ");

System.out.println(s3);

//using concat method

System.out.println("Using concat method: ");

System.out.println(s1.concat(s2));

}

}

**Output of above program is:**

Using + operator:

HelloWorld

Using concat method:

HelloWorld

Check this post for more information about [String Concatenation in Java](https://www.journaldev.com/542/string-concatenation-in-java).

**Java String Pool**

Memory management is the most important aspect of any programming language. Memory management in case of string in java is little bit different than any other class. To make java more memory efficient, JVM introduced a special memory area for string called String Constant Pool.

When we creates a string using string literal it checks if there is identical string already exist in string pool or not. If it is there then it will return the reference of the existing string of string pool.

Let’s have a look at the below example program.

package com.synergetics.string-demo.examples;

/\*\*

\* Java String Pool Example

\*

\*/

public class StringPoolExample {

public static void main(String[] args) {

String a = "abc";

String b = "abc";

String c = "def";

//same reference

if (a==b) {

System.out.println("Both string refer to the same object");

}

//different reference

if (a==c) {

System.out.println("Both strings refer to the same object");

}else {

System.out.println("Both strings refer to the different object");

}

}

}

**Output of above program is:**

Both string refer to the same object

Both strings refer to the different object

Check this post for more about [Java String Pool](https://www.journaldev.com/797/what-is-java-string-pool).

**String intern() Method**

When we creates a string using string literal, it will be created in string pool but what if we create a string using new keyword with same value that exists in string pool? Can we move the String from heap memory to string pool?

For this intern() method is used and it return canonical representation of string object. When we call intern() method on string object that is created using new keyword, it checks if there is already a String with same value in pool?

If yes, then it returns the reference of that String object from pool. If not, then it creates a new String with the same content in the pool and returns the reference.

package com.synergetics.string-demo.examples;

/\*\*

\* Java String intern

\*

\* @author Smita

\*

\*/

public class StringInternExample {

public static void main(String[] args) {

String s1 = "Smita";

String s2 = "Smita";

String s3 = new String("Smita");

System.out.println(s1==s2);//true

System.out.println(s2==s3);//false

String s4 = s3.intern();

System.out.println(s1==s4);//true

}

}

**String Immutability Benefits**

Some of the benefits of String being immutable class are:

1. String Constant Pool, hence saves memory.
2. Security as it’s can’t be changed.
3. Thread safe
4. Class Loading security

Reference: [API Doc](https://docs.oracle.com/javase/9/docs/api/java/lang/String.html)

### Java String to int conversion

Overlooking the exception it can throw, use this:

|  |  |
| --- | --- |
| 1 | int i = Integer.parseInt(myString); |

If the String signified by the variable myString is a valid integer like “1”, “200”, and it will be converted to a Java int. If it fails for any reason, the change can throw a NumberFormatException, so the code should be a little longer to account for this.

### Complete String to int

**In Java, you can use Integer.parseInt() to converta String to int.**

1. **Integer**.parseInt() Examples. Example to **convert**a **String** “10” to an primitive **int**. ...
2. **Integer**.valueOf() Examples. Alternatively, you can use **Integer**.valueOf() , it will returns an **Integer** object. ...
3. NumberFormatException.

Here is the source code for a whole instance program that demonstrates the Java String to int conversion method, control for a possible NumberFormatException:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | package com.synergetics.string-demo.examples;  /\*\*  \* Java String intern  \*  \* @author Smita  \*  \*/  public class JavaStringToIntExample  {      public static void main(String args[])    {  String number = "10";  int result = Integer.parseInt(number);  System.out.println(result);  Integer result1 = Integer.valueOf(number);  System.out.println(result1);    }  } |

**Note**  
In summary, parseInt(String) returns a primitive int, whereas valueOf(String) returns a new Integer() object. If the string does not contain a parsable integer, a NumberFormatException will be thrown.

## References

1. [Integer.parseInt() JavaDoc](http://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html#parseInt(java.lang.String))
2. [Integer.valueOf() JavaDoc](http://docs.oracle.com/javase/7/docs/api/java/lang/Integer.html#valueOf(java.lang.String,%20int))
3. [NumberFormatException JavaDoc](http://docs.oracle.com/javase/7/docs/api/java/lang/NumberFormatException.html)

## String to Float Conversion in Java

There are three main ways to convert a String to float primitive in Java

* Float.parseFloat()
* Float.valueOf()
* new Float()

### Example: Convert using Float.parseFloat()

The Float.parseFloat() static method parses the string argument and returns a float value. The parameter s will be converted to a primitive float value. Note that the method will throw a NumberFormatException if the parameter is not a valid float.

String numberAsString = "153.25";

float number = Float.parseFloat(numberAsString);

System.out.println("The number is: " + number);

### Example: Convert using Float.valueOf()

### The Float.valueOf() static method will return a Float object holding the value of the specified String. Note that the method will throw a NumberFormatException if the parameter is not a valid float.

String numberAsString = "153.25";

float number = Float.valueOf(numberAsString);

System.out.println("The number is: " + number);

### Example: Convert using new Float(String).floatValue()

### Another alternative method is to create an instance of Float class and then invoke it's floatValue() method.

String numberAsString = "153.25";

Float floatObject = new Float(numberAsString);

float number = floatObject.floatValue();

## Java String to Date Example

Let's see the simple code to convert String to Date in java.

1. **import** java.text.SimpleDateFormat;
2. **import** java.util.Date;
3. **public** **class** StringToDateExample1 {
4. **public** **static** **void** main(String[] args)**throws** Exception {
5. String sDate1="31/12/1998";
6. Date date1=**new** SimpleDateFormat("dd/MM/yyyy").parse(sDate1);
7. System.out.println(sDate1+"\t"+date1);
8. }
9. }

## Java Date to String Example

Let's see the simple code to convert Date to String in java.

1. Date date = Calendar.getInstance().getTime();
2. DateFormat dateFormat = **new** SimpleDateFormat("yyyy-mm-dd hh:mm:ss");
3. String strDate = dateFormat.format(date);

Example:

1. **import** java.text.DateFormat;
2. **import** java.text.SimpleDateFormat;
3. **import** java.util.Date;
4. **import** java.util.Calendar;
5. **public** **class** DateToStringExample1 {
6. **public** **static** **void** main(String args[]){
7. Date date = Calendar.getInstance().getTime();
8. DateFormat dateFormat = **new** SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
9. String strDate = dateFormat.format(date);
10. System.out.println("Converted String: " + strDate);
12. }
13. }