**Day 10 Strings**

In Java, a string is a sequence of characters. For example, "hello" is a string containing a sequence of characters 'h', 'e', 'l', 'l', and 'o'.

We use **double quotes** to represent a string in Java. For example,

// create a string

String type = "Java programming";

Here, we have created a string variable named type. The variable is initialized with the string Java Programming.

**Example: Create a String in Java**

class Main {

public static void main(String[] args) {

// create strings

String first = "Java";

String second = "Python";

String third = "JavaScript";

// print strings

System.out.println(first); // print Java

System.out.println(second); // print Python

System.out.println(third); // print JavaScript

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

In the above example, we have created three strings named first, second, and third. Here, we are directly creating strings like primitive types.

However, there is another way of creating Java strings (using the new keyword). We will learn about that later in this tutorial.

**Note**: Strings in Java are not primitive types (like int, char, etc). Instead, all strings are objects of a predefined class named String.

And, all string variables are instances of the String class.

**Java String Operations**

Java String provides various methods to perform different operations on strings. We will look into some of the commonly used string operations.

**1. Get length of a String**

To find the length of a string, we use the length() method of the String. For example,

class Main {

public static void main(String[] args) {

// create a string

String greet = "Hello! World";

System.out.println("String: " + greet);

// get the length of greet

int length = greet.length();

System.out.println("Length: " + length);

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Output**

String: Hello! World

Length: 12

In the above example, the length() method calculates the total number of characters in the string and returns it. To learn more, visit [Java String length()](https://www.programiz.com/java-programming/library/string/length).

**2. Join Two Java Strings**

We can join two strings in Java using the concat() method. For example,

class Main {

public static void main(String[] args) {

// create first string

String first = "Java ";

System.out.println("First String: " + first);

// create second

String second = "Programming";

System.out.println("Second String: " + second);

// join two strings

String joinedString = first.concat(second);

System.out.println("Joined String: " + joinedString);

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Output**

First String: Java

Second String: Programming

Joined String: Java Programming

In the above example, we have created two strings named first and second. Notice the statement,

String joinedString = first.concat(second);

Here, the concat() method joins the second string to the first string and assigns it to the joinedString variable.

We can also join two strings using the + operator in Java. To learn more, visit [Java String concat()](https://www.programiz.com/java-programming/library/string/concat).

**3. Compare two Strings**

In Java, we can make comparisons between two strings using the equals() method. For example,

class Main {

public static void main(String[] args) {

// create 3 strings

String first = "java programming";

String second = "java programming";

String third = "python programming";

// compare first and second strings

boolean result1 = first.equals(second);

System.out.println("Strings first and second are equal: " + result1);

// compare first and third strings

boolean result2 = first.equals(third);

System.out.println("Strings first and third are equal: " + result2);

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Output**

Strings first and second are equal: true

Strings first and third are equal: false

In the above example, we have created 3 strings named first, second, and third. Here, we are using the equal() method to check if one string is equal to another.

The equals() method checks the content of strings while comparing them. To learn more, visit [Java String equals()](https://www.programiz.com/java-programming/library/string/equals).

**Note**: We can also compare two strings using the == operator in Java. However, this approach is different than the equals() method. To learn more, visit [Java String == vs equals()](https://www.programiz.com/java-programming/examples/differentiate-string-equals).

**Escape character in Java Strings**

The escape character is used to escape some of the characters present inside a string.

Suppose we need to include double quotes inside a string.

// include double quote

String example = "This is the "String" class";

Since strings are represented by **double quotes**, the compiler will treat "This is the " as the string. Hence, the above code will cause an error.

To solve this issue, we use the escape character \ in Java. For example,

// use the escape character

String example = "This is the \"String\" class.";

Now escape characters tell the compiler to escape **double quotes** and read the whole text.

**Java Strings are Immutable**

In Java, strings are **immutable**. This means, once we create a string, we cannot change that string.

To understand it more deeply, consider an example:

// create a string

String example = "Hello! ";

Here, we have created a string variable named example. The variable holds the string "Hello! ".

Now suppose we want to change the string.

// add another string "World"

// to the previous tring example

example = example.concat(" World");

Here, we are using the concat() method to add another string World to the previous string.

It looks like we are able to change the value of the previous string. However, this is not true.

Let's see what has happened here,

1. JVM takes the first string "Hello! "
2. creates a new string by adding "World" to the first string
3. assign the new string "Hello! World" to the example variable
4. the first string "Hello! " remains unchanged

**Creating strings using the new keyword**

So far we have created strings like primitive types in Java.

Since strings in Java are objects, we can create strings using the new keyword as well. For example,

// create a string using the new keyword

String name = new String("Java String");

In the above example, we have created a string name using the new keyword.

Here, when we create a string object, the String() constructor is invoked. To learn more about constructor, visit [Java Constructor](https://www.programiz.com/java-programming/constructors).

**Note**: The String class provides various other constructors to create strings. To learn more, visit [Java String (official Java documentation)](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html).

**Example: Create Java Strings using the new keyword**

class Main {

public static void main(String[] args) {

// create a string using new

String name = new String("Java String");

System.out.println(name); // print Java String

}

}

[Run Code](https://www.programiz.com/java-programming/online-compiler)

**Create String using literals vs new keyword**

Now that we know how strings are created using string literals and the new keyword, let's see what is the major difference between them.

In Java, the JVM maintains a **string pool** to store all of its strings inside the memory. The string pool helps in reusing the strings.

1. While creating strings using string literals,

String example = "Java";

Here, we are directly providing the value of the string (Java). Hence, the compiler first checks the string pool to see if the string already exists.

* **If the string already exists**, the new string is not created. Instead, the new reference, example points to the already existed string (Java).
* **If the string doesn't exist**, the new string (Java is created.

2. While creating strings using the new keyword,

String example = new String("Java");

Here, the value of the string is not directly provided. Hence, a new "Java" string is created even though "Java" is already present inside the memory pool.

**Methods of Java String**

Besides those mentioned above, there are various [string methods](https://www.programiz.com/java-programming/library/string) present in Java. Here are some of those methods:

|  |  |
| --- | --- |
| Methods | Description |
| [contains()](https://www.programiz.com/java-programming/library/string/contains) | checks whether the string contains a substring |
| [substring()](https://www.programiz.com/java-programming/library/string/substring) | returns the substring of the string |
| [join()](https://www.programiz.com/java-programming/library/string/join) | join the given strings using the delimiter |
| [replace()](https://www.programiz.com/java-programming/library/string/replace) | replaces the specified old character with the specified new character |
| [replaceAll()](https://www.programiz.com/java-programming/library/string/replaceall) | replaces all substrings matching the regex pattern |
| [replaceFirst()](https://www.programiz.com/java-programming/library/string/replacefirst) | replace the first matching substring |
| [charAt()](https://www.programiz.com/java-programming/library/string/charat) | returns the character present in the specified location |
| [getBytes()](https://www.programiz.com/java-programming/library/string/getbytes) | converts the string to an array of bytes |
| [indexOf()](https://www.programiz.com/java-programming/library/string/indexof) | returns the position of the specified character in the string |
| [compareTo()](https://www.programiz.com/java-programming/library/string/compareto) | compares two strings in the dictionary order |
| [compareToIgnoreCase()](https://www.programiz.com/java-programming/library/string/comparetoignorecase) | compares two strings ignoring case differences |
| [trim()](https://www.programiz.com/java-programming/library/string/trim) | removes any leading and trailing whitespaces |
| [format()](https://www.programiz.com/java-programming/library/string/format) | returns a formatted string |
| [split()](https://www.programiz.com/java-programming/library/string/split) | breaks the string into an array of strings |
| [toLowerCase()](https://www.programiz.com/java-programming/library/string/tolowercase) | converts the string to lowercase |
| [toUpperCase()](https://www.programiz.com/java-programming/library/string/touppercase) | converts the string to uppercase |
| [valueOf()](https://www.programiz.com/java-programming/library/string/valueof) | returns the string representation of the specified argument |
| [toCharArray()](https://www.programiz.com/java-programming/library/string/tochararray) | converts the string to a char array |
| [matches()](https://www.programiz.com/java-programming/library/string/matches) | checks whether the string matches the given regex |
| [startsWith()](https://www.programiz.com/java-programming/library/string/startswith) | checks if the string begins with the given string |
| [endsWith()](https://www.programiz.com/java-programming/library/string/endswith) | checks if the string ends with the given string |
| [isEmpty()](https://www.programiz.com/java-programming/library/string/isempty) | checks whether a string is empty of not |
| [intern()](https://www.programiz.com/java-programming/library/string/intern) | returns the canonical representation of the string |
| [contentEquals()](https://www.programiz.com/java-programming/library/string/contentequals) | checks whether the string is equal to charSequence |
| [hashCode()](https://www.programiz.com/java-programming/library/string/hashcode) | returns a hash code for the string |
| [subSequence()](https://www.programiz.com/java-programming/library/string/subsequence) | returns a subsequence from the string |