Strings in Java

Introduction

String is a sequence of characters.

In the Java programming language, strings are objects.

Java provides a class called “String” in “java.lang” package to create and manipulate strings.

The string value must be represented in “ ” (double cotes).

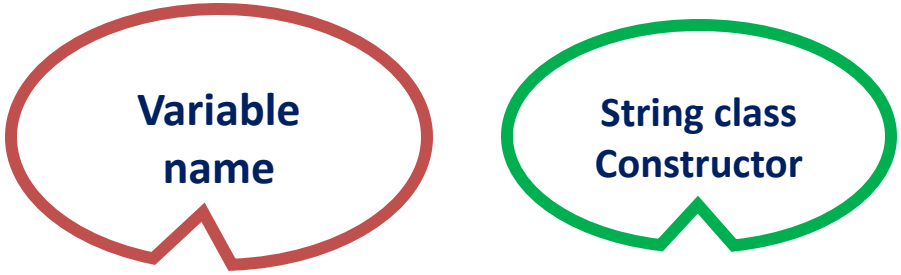
Creating String

Direct way to create a string


```
String  greeting = “Good Morning!";
```

Creating String

Another way to create a string




```
String  greeting = new String("Good Morning!");
```

Creating String

Another way to create a string

```
char  msg[ ] = {"J,'A','V','A'};
String  name = new String(msg);
```

Here, first we create a character array and pass that array as argument to the string class constructor

Operations in String Class

length()

concat()

char charAt(int index)

int compareTo(String anotherString)

int compareToIgnoreCase(String str)

boolean endsWith(String suffix)

boolean equals(Object anObject)

boolean equalsIgnoreCase(String anotherString)

Operations in String Class

int indexOf(int ch)

int indexOf(int ch, int fromIndex)

int indexOf(String str)

int lastIndexOf(int ch)

int lastIndexOf(int ch, int fromIndex)

int lastIndexOf(String str)

boolean startsWith(String prefix, int toffset)

String substring(int beginIndex, int endIndex)

boolean matches(String regex)

boolean startsWith(String prefix)

Operations in String Class

String substring(int beginIndex)

String toLowerCase()

String toString()

String toUpperCase()

String trim()

length()

This method returns a integer which is equal to total number of Characters present in a string object.

For example, the string length of "Note Book" is 9 including the space character.

Syntax:

```
stringVariable.length();
```

Example:

```
greeting.length();
```

concat()

String concatenation is the operation of joining two string objects end to end.
For example, the concatenation of two strings "Foot” and "ball” is "Football“

Syntax:

```
stringVariable1. concat(stringVariable2);
```

Example:

```
greeting.concat(name);
```

concat()

String concatenation can also be done using + operator
For example, the concatenation of two strings "Foot” and "ball” is "Football“ as follows

String game = “Foot”+”ball”;

char charAt(int index)

This method takes an index as a parameter and returns the character that is present at that index of the string.

Syntax:

```
stringVariable. charAt(IndexValue);
```

Example:

```
greeting.charAt(2);
```

equals

The equals is used to compare two Strings and states whether they are equal or not.

Syntax:

```
stringVariable1. equals(stringVariable2);
```

Example:

```
greeting.equals(name);
```

equalsIgnoreCase

The equals IgnoreCase is used to compare two Strings by ignoring letter case and states whether they are equal or not.

Syntax:

stringVariable1. equalgnoreCase(stringVariable2);

Example:

```
greeting.equalsIgnoreCase(name);
```

startsWith()

The startsWith accepts a string and checks whether the invoking string starts with the given string or not.

Syntax:

```
stringVariable1. startsWith(stringVariable2);
```

Example:

```
greeting.startsWith(name);
```

startsWith()

Note: startsWith method also have one more form as follows

Syntax:

```
stringVariable1. startsWith(stringVariable2,Index);
```

Example:

```
greeting.startsWith(name,10);
```

endsWith()

The endsWith accepts a string and determines whether the invoking string ends with the given string or not.

Syntax:

```
stringVariable1. endsWith(stringVariable2);
```

Example:

```
greeting.endsWith(name);
```

Searching Strings

The String class provides 2 methods for searching a string.

indexOf() : Searches for the first occurrence of a character or substring.

lastIndexOf() : Searches for the last occurrence of a character or substring

Syntax:

```
stringVariable.indexOf(charVariable);
stringVariable.lastIndexOf(charVariable);
stringVariable.indexOf(stringVariable);
stringVariable.indexOf(charVariable, startIndex);
stringVariable.lastIndexOf(charVariable, startIndex);
```

substring - Modifying a String

The substring method can be used to extract some characters from the String.
This method comes in two forms

Syntax:

```
stringVariable1 = stringVariable2.substring(index);

stringVariable1 = stringVariable2.substring(index1, index2);
```

replace

The replace() is used to replace a character or a sequence of characters of an invoking string with a desired character or set of characters

Syntax:

```
stringVariable.replace(originalChar, replcedChar);
```

trim

trim is used to trim/remove whitespace from the beginning and the end of the string that invoked this method.

Syntax:

```
stringVariable.trim();
```