Advanced Functional Thinking

Dr Surbhi Saraswat
School of Computer Science

Scala Strings

Creating a String

```
val greeting: String = "Hello, world!"
println( greeting )
```

String Length

```
var palindrome = "Dot saw I was Tod";
var len = palindrome.length();
println( "String Length is : " + len );
```

String Concatenation

```
var str1 = "Dot saw I was ";
var str2 = "Tod";
println("Dot " + str1 + str2);
```

Scala Strings

String Concatenation

The String class includes a method for concatenating two strings −

```
string1.concat(string2);
```

This returns a new string that is string1 with string2 added to it at the end.

You can also use the concat() method with string literals, as in −

```
"My name is ".concat("Zara");
```

Scala Strings

String Interpolation

String Interpolation allows users to embed variable references directly in the processed string literals.

```
var stringVar = "Scala!"
println(s"Hello, $stringVar")
```

String interpolation is nothing but variable substitution with its value inside a string.

Generally Variable interpolation occurs when the string literal is double-quoted.

The variables are recognized because variables start with a sigil (typically "$").

Scala Strings

String Interpolation

```
Example: s"a\nb"

res0: String = a

          B

Example: raw"a\nb"

res1: String = a\nb
```

The **raw** String Interpolator: - The raw interpolator is similar to the **s** interpolator except that it performs “No escaping of literals within the string”.

In the first example: s interpolator is used.

In second example : The raw interpolator is useful when you want to avoid having a sequence of characters like \n turn into a newline character.

Scala Strings

String Methods

These are the most important `String` methods (here, `S` and `T` are strings):

- `S.length` is the length of the string in characters;
- `S.substring(i)` returns the part of the string starting at index `i`.
- `S.substring(i,j)` returns the part of the string starting at index `i` and going up to index `j-1`. You can write `S.slice(i,j)` instead.
- `S.contains(T)` returns `true` if `T` is a substring of `S`;
- `S.indexOf(T)` returns the index of the first occurrence of the substring `T` in `S` (or -1);
- `S.indexOf(T, i)` returns the index of the first occurrence after index *i* of the substring `T` in `S` (or -1);

Scala Strings

String Methods

These are the most important `String` methods (here, `S` and `T` are strings):

- `S.toLowerCase` and `S.toUpperCase` return a copy of the string with all characters converted to lower or upper case;
- `S.capitalize` returns a new string with the first letter only converted to upper case;
- `S.reverse` returns the string backwards;
- `S.isEmpty` is the same as `S.length == 0`;
- `S.nonEmpty` is the same as `S.length != 0`;
- `S.startsWith(T)` returns `true` if `S` starts with `T`;
- `S.endsWith(T)` returns `true` if `S` ends with `T`;

Scala Strings

String Methods

These are the most important `String` methods (here, `S` and `T` are strings):

- `S.replace(c1, c2)` returns a new string with all characters `c1` replaced by `c2`;
- `S.replace(T1, T2)` returns a new string with all occurrences of the substring `T1` replaced by `T2`;
- `S.trim` returns a copy of the string with white space at both ends removed;
- `S.split(T)` splits the string into pieces and returns an array with the pieces.

Scala Strings

String Methods

List of string methods defined by the class java.lang.String.

1. char charAt(int index)

This method returns the character at the index we pass to it. Isn’t it so much like Java?

```
scala> "Ayushi".charAt(1)
res2: Char = y
```

2. String trim()

Removes leading and trailing whitespace frome string.

```
scala> "    Ayushi      ".trim
res10: String = Ayushi
```

2. int compareTo(String anotherString)

This is like the previous one, except that it compares two strings lexicographically. If they match, it returns 0.

Otherwise, it returns the difference between the two(the number of characters less in the shorter string, or the maximum ASCII difference between the two).

```
scala> "Ayushi".compareTo("Ayushi ")
res3: Int = -1
scala> "Ayushi".compareTo("Ayushi")
res4: Int = 0
scala> "Ayushi".compareTo("ayushi")
res5: Int = -32
```

Scala Strings

Comparing to a different type raises a type-mismatch error.

```
scala> "Ayushi".compareTo(7)
<console>:12: error: type mismatch;
found : Int(7)    required: String
```

4. int compareToIgnoreCase(String str)

compares two strings lexicographically, while ignoring the case differences.

```
scala> "Ayushi".compareToIgnoreCase("ayushi")
res21: Int = 0
scala> "Ayushi".compareToIgnoreCase("ayushi ")
res22: Int = -1
scala> "Ayushi".compareToIgnoreCase("aYushi")
res41: Int = 0
```

5. String concat(String str)

This will concatenate the string in the parameter to the end of the string on which we call it. We saw this when we talked about strings briefly.

```
scala> "Ayushi".concat("Sharma")
res23: String = AyushiSharma
```

6. Boolean equals(String str)

If equal, it returns true; otherwise, false.

```
scala> "Ayushi".equals("Ayushi")
res24: Boolean = true
scala> "ayushi".equals("Ayushi")
res25: Boolean = false
```

Scala Strings

7. Boolean endsWith(String suffix)

```
scala> "Ayushi".endsWith("i")
res32: Boolean = true
scala> "Ayushi".endsWith("sha")
res33: Boolean = false
```

8. Boolean startsWith(String suffix)

```
scala> "Ayushi".startsWith(“A")
res32: Boolean = true
scala> "Ayushi".startsWith("a")
res33: Boolean = false
```

9. String replace(“s1”, “s2”)

```
val str = "Hello, World!"
println(str.replace("Hello", "Hi"))
res34: "Hi, World!”
```

10. String substring(i, j)

```
val str = "Hello, World!"
println(str.substring(3,7))        // lo, W
println(str.substring(3))          // lo, World!
```