Kotlin String

The String class represents an array of char types.

Strings are immutable which means the length and elements cannot be changed after their creation.

```
val ch = charArrayOf('h', 'e', 'l', 'l', 'o')
val st = String(ch)
```

Unlike Java, Kotlin does not require a new keyword to instantiate an object of a String class.
A String can be simply declared within double quote (" ") known as escaped string or triple quote(""" """) known as raw string.

```
val str1 = "Hello, javaTpoint"
val str2 = """Welcome To JavaTpoint"""
```

Kotlin String Property

Property	Description
length: Int	It returns the length of string sequence.
indices: IntRange	It returns the ranges of valid character indices from current char sequence.
lastIndex: Int	It returns the index of last character from char sequence.

String Function

Functions	Description
compareTo(other: String): Int	It compares the current object with specified object for order. It returns zero if current is equals to specified other object.
get(index: Int): Char	It returns the character at given index from the current character sequence.
plus(other: Any?): String	It returns the concatenate string with the string representation of the given other string.
subSequence(startIndex: Int,endIndex: Int): CharSequence	It returns the new character sequence from current character sequence, starting from startIndex to endIndex.
CharSequence.contains(other: CharSequence, ignoreCase: Boolean = false):Boolean	It returns true if the character sequence contains the other specified character sequence.
CharSequence.count(): Int	It returns the length of char sequence.
String.drop(n: Int): String	It returns a string after removing the first n character.
String.dropLast(n: Int): String	It returns a string after removing the last n character.
String.dropWhile (predicate: (Char) -> Boolean): String	It returns a character sequence which contains all the characters, except first characters which satisfy the given predicate.
CharSequence.elementAt(index: Int): Char	It returns a character at the given index or throws an IndexOutOfBoundsException if the index does not exist in character sequence.
CharSequence.indexOf(char: Char, startIndex: Int = 0, ignoreCase: Boolean = false): Int	It returns the index of first occurrence of the given character, starting from the given index value.
CharSequence.indexOfFirst(predicate: (Char) -> Boolean): Int	It returns the index of first character which match the given predicate, or -1 if the character sequence not contains any such character.
CharSequence.indexOfLast(predicate: (Char) -> Boolean	It returns the index of last character which match the given predicate, or -1

): Int	if the character sequence not contains any such character.
CharSequence.getOrElse(index: Int, defaultValue: (Int) ->Char): Char	It returns the character at specified index or the result of calling the defaultValue function if the index is out of bound of current character sequence.
CharSequence.getOrNull(index: Int): Char?	It returns a character at the given index or returns null if the index is out of bound from character sequence.

String elements and templates

String elements

- The characters which are present in string are known as elements of string.
- Element of string are accessed by indexing operation string[index].
- String's index value starts from 0 and ends at one less than the size of string string[string.length-1].
- Index 0 represent first element, index 1 represent second element and so on.

```
val str ="Hello, javatpoint"
   println(str[0]) //prints H
```

```
fun main(args: Array<String>) {
     val str = "Hello, javatpoint!"
     println(str[0])          // --> H
     println(str[1])          // --> e
     println(str[str.length-1])    // --> !
}
```

String templates

String template expression is a piece of code which is evaluated and its result is returned into string.
Both string types (escaped and raw string) contain template expressions.
String templates starts with a dollar sign $ which consists either a variable name or an arbitrary expression in curly braces.

String template as variable name:

```
val i =10
print("i = $i") //i=10

fun main(args: Array<String>) {
     val i =10
     print("i = $i")//i=10
}
```

String template as arbitrary expression in curly braces:

String template is also used in arbitrary expression in curly braces to evaluate a string expression. This is done by using dollar sign $.

```
fun main(args: Array<String>) {
     val str = "abc"
     println("$str is a string which length is
               ${str.length}")
}
```

String template in raw string:

```
fun main(args: Array<String>) {
val a = 10
val b = 5
val myString = """value $a
               |is greater than value $b
               """.trimMargin()
     println("${myString.trimMargin()}")
}
```

Kotlin String Literals

Kotlin has two types of string literals:

- Escaped String
- Raw String

Escaped String

Escape String is declared within double quote (" ") and may contain escape characters like '\n', '\t', '\b' ,'\r','\$'etc.

```
val text1 ="Hello, JavaTpoint"
//or
val text2 ="Hello, JavaTpoint\n"
//or
val text3 ="Hello, \nJavaTpoint"
```

Raw String

- Row String is declared within triple quote (""" """).
- It provides facility to declare String in new lines and contain multiple lines.
- Row String cannot contain any escape character.

```
val text1 ="""
            Welcome
              To
                JavaTpoint
           """
```

- While using raw string with new line, it generates a | as margin prefix.

For example:

```
fun main(args: Array<String>) {
val text = """Kotlin is official language
               |announce by Google for
               |android application development
           """
           println(text)
}
```

Output:

```
Kotlin is official language
        |announce by Google for
        |android application development
```

String trimMargin() function

Leading whitespace can be removed with trimMargin() function. By default, trimMargin() function uses | as margin prefix.

```
fun main(args: Array<String>) {
val text = """Kotlin is official language
        |announce by Google for
        |android application development
    """.trimMargin()
     println(text)
}
```

Output:

```
Kotlin is official language
announce by Google for
android application development
```

However, it can be change by passing a new string inside trimMargin() function.

```
fun main(args: Array<String>) {

val text = """Kotlin is official language
        #announce by Google for
        #android application development
    """.trimMargin("#")
     println(text)
}
```

Output:

```
Kotlin is official language
announce by Google for
android application development
```

Kotlin String Equality

In Kotlin, strings equality comparisons are done on the basis of structural equality (==) and referential equality (===).

In structural equality two objects have separate instances in memory but contain same value.

Referential equality specifies that two different references point the same instance in memory.

Structural equality (==)

To check the two objects containing the same value,

we use == operator or != operator for negation.

It is equivalent to *equals()* in java.

```
fun main(args: Array<String>) {
    val str1 = "Hello, World!"
    val str2 = "Hello, World!"
    println(str1==str2) //true
    println(str1!=str2) //false
}
```

Referential equality (===)

To check the two different references point to the same instance, we use === operator.

The !== operator is used for negation.

a === b specifies true if and only if a and b both point to the same object.

Let's see an example of referential equality to check different reference contains same instance or not.

For creating string we are using a helper method buildString rather than using quotes.

```
fun main(args: Array<String>) {
    val str1 = buildString { "string value" }
    val str2 = buildString { "string value" }
    println(str1===str2)  // false
    println(str1!==str2)  // true
}
```