Core APIs

Dates & Times

String Methods

String concatenation

1. If both operands are numeric, + means numeric addition.
2. If either operand is a String, + means concatenation.
3. The expression is evaluated left to right.

String strip

Whitespace consists of spaces along with the `\t` (tab) and `\n` (newline) characters.

```
String s = "  01234   ";
//remove leading & trailing spaces
var s1 = s.strip();  // "01234"

//remove leading spaces
var s2=s.stripLeading();  // "01234   "

//remove trailing spaces
var s3 = s.stripTrailing(); // "  01234"
```

CharSequence

Both **String** and **StringBuilder** implement *CharSequence*.

formatted

```
jshell> var name = "John"
name ==> "John"

jshell> var age = 30
age ==> 30

jshell> var s = "%s is %d years old".formatted(name, age)
s ==> "John is 30 years old"
```

String pool

```
String s1 = "ab";
String s2 = "a"+"b";
System.out.println(s1==s2);  //true
String s3 = "a";
s3 +="b";
System.out.println(s1==s3);  //false
```

StringBuilder

setLength

```
var sb = new StringBuilder("0123456789");
sb.setLength(5);
System.out.println(sb);  //01234
```

append

There ain't method such as: ~~append(index, string)~~

```
var sb = new StringBuilder("hello");
//append has only a single argument
sb.append(" World");
```

StringBuilder Methods

Arrays

```
jshell> int[] array = new int[]{1,2,3}
array ==> int[3] { 1, 2, 3 }

jshell> int[] array = {1,2,3}  //anonymous array
array ==> int[3] { 1, 2, 3 }
```

Arrays declaration/creation

```
java.util.Date[] dates[] = new java.util.Date[2][];
```

```
int[] array = new int[3];  //here I specify the dimenstion
int[] array2 = new int[]{1, 4}; //here I specify the values

// array creation with both dimension expression and initialization
//int[] array3 = new int[2]{1,2}; //does not compile!
```

Multi dimensional array

```
int[][] a3D[], a2D;
a3D = new int[2][2][2];  //3D array
a2D = new int[3][3];  //2D array



int[][] matrix = new int[][]{{1,2}, {3}};
int[][] matrix2 = new int[2][1];
int[][] matrix3 = {{1,2}, {3}};

//int[][] matrix4 = new int[2][1]{{1,2}, {3}};  //does not compile
```

Multi Dimensional Array

mismatch

```
String[] s1 = {"Camel", "Peacock", "Llama"};
String[] s2 = {"Camel", "Llama", "Peacock"};
String[] s3 = {"Camel", "Llama", "Peacock"};
int position = Arrays.mismatch(s1, s2); //first element to differ i
System.out.println(position);     //1
position = Arrays.mismatch(s2, s3); //they are identical then retur
System.out.println(position);     //-1
```

compare

```
int[] array1 = {1, 3, 5};
int[] array2 = {1, 3, 5};
int[] array3 = {1, 5, 5};
int result = Arrays.compare(array1, array2);
System.out.println(result);  //0  (the 2 arrays are identical)

//the first element which they differ is the 2nd position (3 vs 5)
//In this case, we get a negative number because 3 is smaller than 
result = Arrays.compare(array1, array3);
System.out.println(result);  //-1
```

Arrays Sort String (natural order)

1. Numbers
2. Uppercase
3. Lowercase

Binary Search

```
var arr = new String[]{"PIG", "pig", "123"};
//Numbers sort before letters and uppercase sorts before lowercase.
Arrays.sort(arr);  //"123", "PIG", "pig"
int position = Arrays.binarySearch(arr, "Pippa");
//result : -insertionPoint -1
//insertionPoint (ideally) would be 2 (after "PIG")
//-> result = -2 -1 = -3
System.out.println(position);  //-3
```

Math

floor & ceil

```
double floor = Math.floor(3.8); // 3.0
double ceil = Math.ceil(3.8); // 8.0
```

round

return type:

- `round(double)` is `long`
- `round(float)` is `int`

```
long round = Math.round(5.2D); //5
int round2 round = Math.round(5.2F); //5
```

pow

```
double pow = Math.pow(4.0, 2.0); //16.0
```

Converting String to number

```
//  this method returns a primitive long
var aLongPrimitive = Long.parseLong("123");

//this method returns a Long object
var aLong = Long.valueOf("123");
sout.println(aLong.toString());
```

Manipulating String

indent

However, indent() also normalizes whitespace characters. First, a line break is added to the end of the string if not already there. Second, any line breaks are converted to the \n format.

```
var name = "1234 \n567";
var indent1 = name.indent(1);
System.out.println("#%s#".formatted(indent1));
```

It prints:

```
# 1234
 567
#
```

indent

translateEscapes

\n - new line

```
String source = "Today is\\nSaturday";
System.out.println(source); //Today is\nSaturday

//Today is
//Saturday
System.out.println(source.translateEscapes());
```

\t - tab

```java
String source = "1\\t2";
//  1\t2
System.out.println(source);
//  1 2
System.out.println(source.translateEscapes());
```

translateEscapes