CODEGYM

Java Developer in 12 months

Mentor-supported training program from CodeGym

MODULE 1. JAVA SYNTAX

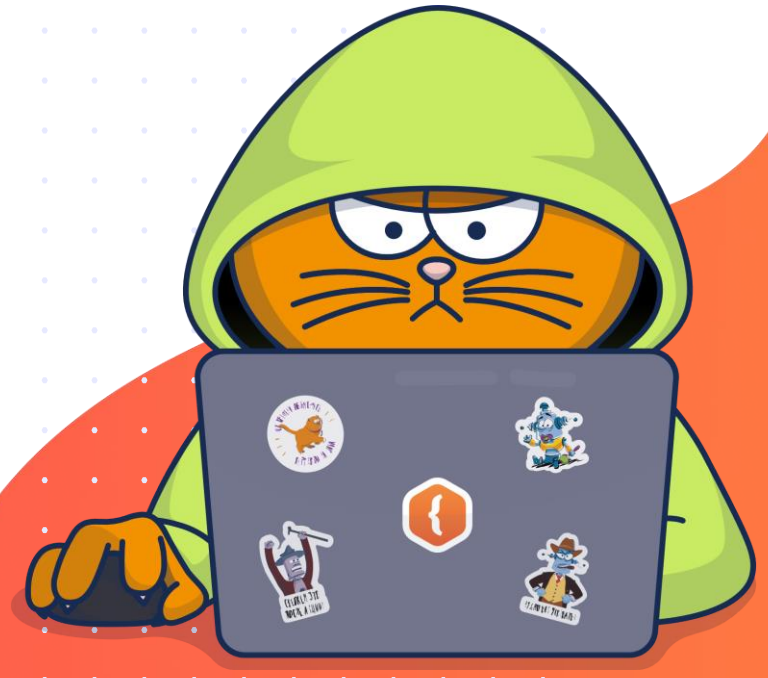
Lesson 10

Elective

CODEGYM

Lesson plan

- Literals in Java
- Numeral systems
- Encoding
- StringTokenizer, StringBuilder, String.format



CODEGYM

Literals in Java

Literals are values that are explicitly set in the program code, i.e. constants of a specific type that are in the code when it is started.

We're not talking about any old data,
but the values of primitive types and the String type.

Code	Literals

`int a = 5;`
`int b = a + 10;`
`String s = "Sum=" + (a + b);`
	`5`
	`10`
	`"Sum="`

The literals in this code are the number 5,
the number 10 and the string 'Sum = '.

All literals are primitive values (numbers, symbols, booleans) and strings.

You cannot create an object literal.

The only object literal that exists is null.

CODEGYM

Numeral systems

In everyday life, we use decimal notation: all our numbers are represented using 10 symbols: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**.

There are 10 numerals, so the system is called decimal.

But programmers are big-time inventors. They immediately came up with encodings that use a different number of symbols. For example, **64, 16, 8** and **2**.

CODEGYM

Unicode encoding

Each character displayed on the screen corresponds to a specific numerical code. A standardized set of these codes is called an encoding.

The first encoding contained only 128 characters. This encoding was called ASCII.

Thus, in 1993, the Unicode encoding was created, and the Java language became the first programming language that used this encoding as the standard for storing text. Now Unicode is the standard for the entire IT industry.

For example, the Unicode character **U+0048** is the uppercase English letter **H**.

```
Hello
U+0048 U+0065 U+006C U+006C U+006F
```

CODEGYM

StringTokenizer class

The class has a constructor and two important methods. We pass the constructor a string that we split into parts, and a string comprised of a set of delimiting characters.

Methods	Description
String nextToken()	Returns the next substring
boolean hasMoreTokens()	Checks whether there are more substrings.

You can create a StringTokenizer object with this command:

```
StringTokenizer name = new StringTokenizer(string, delimiters);
```

Where **string** is the string to be divided into parts.
And **delimiters** is a string, and each character in it is treated as a delimiter.

CODEGYM

StringBuilder

The String class cannot have descendants (final) and instances of the class cannot be changed after creation (immutable). That's precisely why a String-like type that can be changed was added to the Java language. It is called StringBuilder.

To create a StringBuilder object based on an existing string:

```
StringBuilder name = new StringBuilder(string);
```

To create an empty mutable string:

```
StringBuilder name = new StringBuilder();
```

CODEGYM

List of methods of the StringBuilder class

Method	Description
StringBuilder append(obj)	Converts the passed object to a string and appends it to the current string
StringBuilder insert(int index, obj)	Converts the passed object to a string and inserts it into the current string
StringBuilder replace(int start, int end, String str)	Replaces the part of the string specified by the start..end interval with the passed string
StringBuilder deleteCharAt(int index)	Removes the character with the specified index from the string
StringBuilder delete(int start, int end)	Removes characters within the specified interval from the string
int indexOf(String str, int index)	Searches for a substring in the current string
int lastIndexOf(String str, int index)	Searches for a substring in the current string, starting from the end
char charAt(int index)	Returns the character in the string at the passed index
String substring(int start, int end)	Returns the substring defined by the specified interval
StringBuilder reverse()	Reverses the current string.
void setCharAt(int index, char)	Changes the character at the specified index to the passed character
int length()	Returns the length of the string in characters

CODEGYM

String.format() method

The String class has a static format() method: it lets you specify a pattern for assembling a string with data.

The general appearance of the command is as follows:

```
String name = String.format(pattern, parameters);
```

Example:

Code	Result
`String.format("Fullname=%s", name);`	Fullname=Diego

Short list of format specifiers that can be used inside the format string:

Specifier	Meaning
%s	String
%d	integer: byte, short, int, long
%f	real number: float, double
%b	boolean
%c	char
%t	Date
%%	% character

CODEGYM

Arrays — a helper class

Arrays.toString(name)

The Arrays.toString() method returns a string representation of a one-dimensional array, separating the elements with a comma. Instead of running through arrays using a for loop, you can use this method to print elements on the console..

Arrays.deepToString(name)

Java's Arrays.deepToString() method returns a string representation of a multidimensional array, delimiting each nesting level with square brackets.

Arrays.equals(name1, name2)

The method returns true if the arrays are of equal length and their elements are equal. Otherwise, it returns false.

Arrays.deepEquals(name1, name2)

The method returns true if the **multidimensional** arrays are of equal length and their elements are equal. Otherwise, it returns false. If the elements inside the array are also arrays, then the Arrays.deepEquals() method is used to compare them, and so on.

Arrays.fill(name, value)

It fills the passed array with the passed value. It only works with one-dimensional arrays.

Arrays.copyOf(name, length)

This method does not change the existing array, but instead creates a new array and copies the elements of the old array into it. If the elements don't fit (the length is less than the length of the existing array), then the extra values are ignored. If the length of the new array is greater than the length of the old one, the cells are filled with default values.

Arrays.copyOfRange(name, first, last)

This method creates a new array and fills it with elements from an arbitrary place in the original array. Where first and last are the indices of the first and last elements that should be put into the new array. The last element is not included in the range.

Arrays.sort(name)

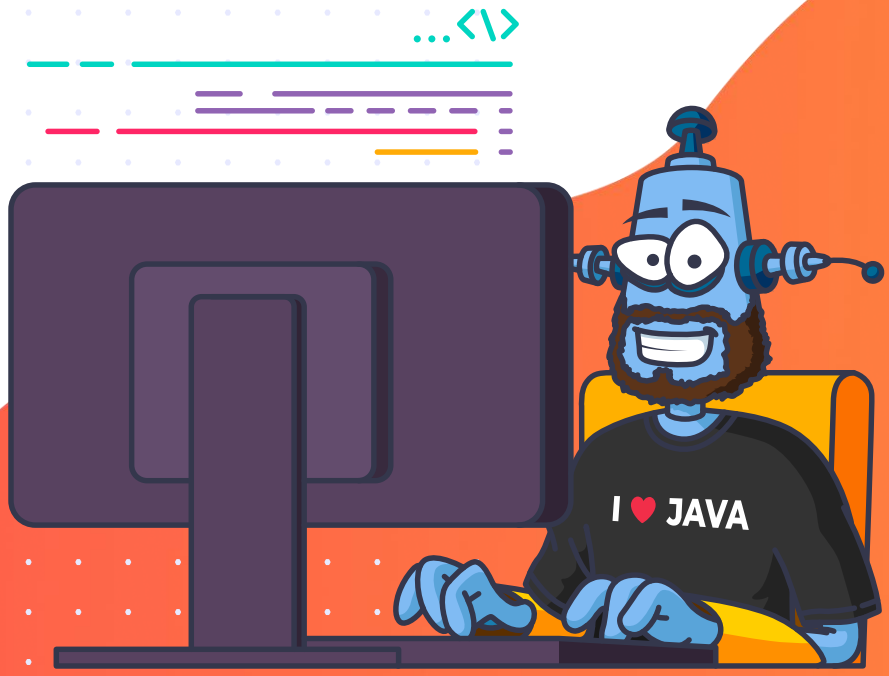
It allows sorting array elements in ascending order

CODEGYM

Homework

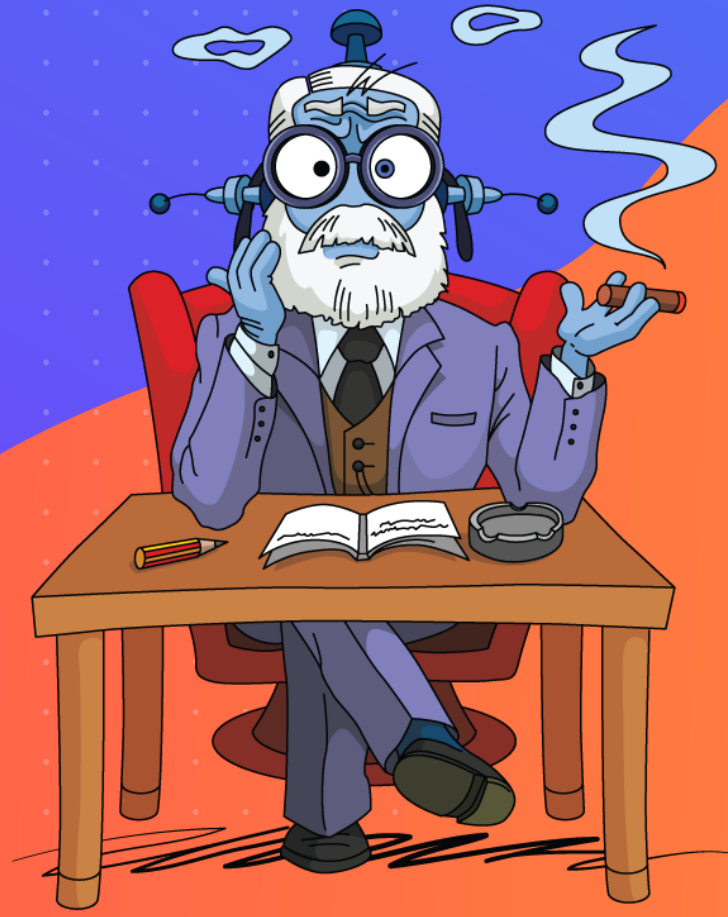
MODULE 1. JAVA SYNTAX

Complete Level 11



CODEGYM

Answers to questions



MODULE 1. JAVA SYNTAX