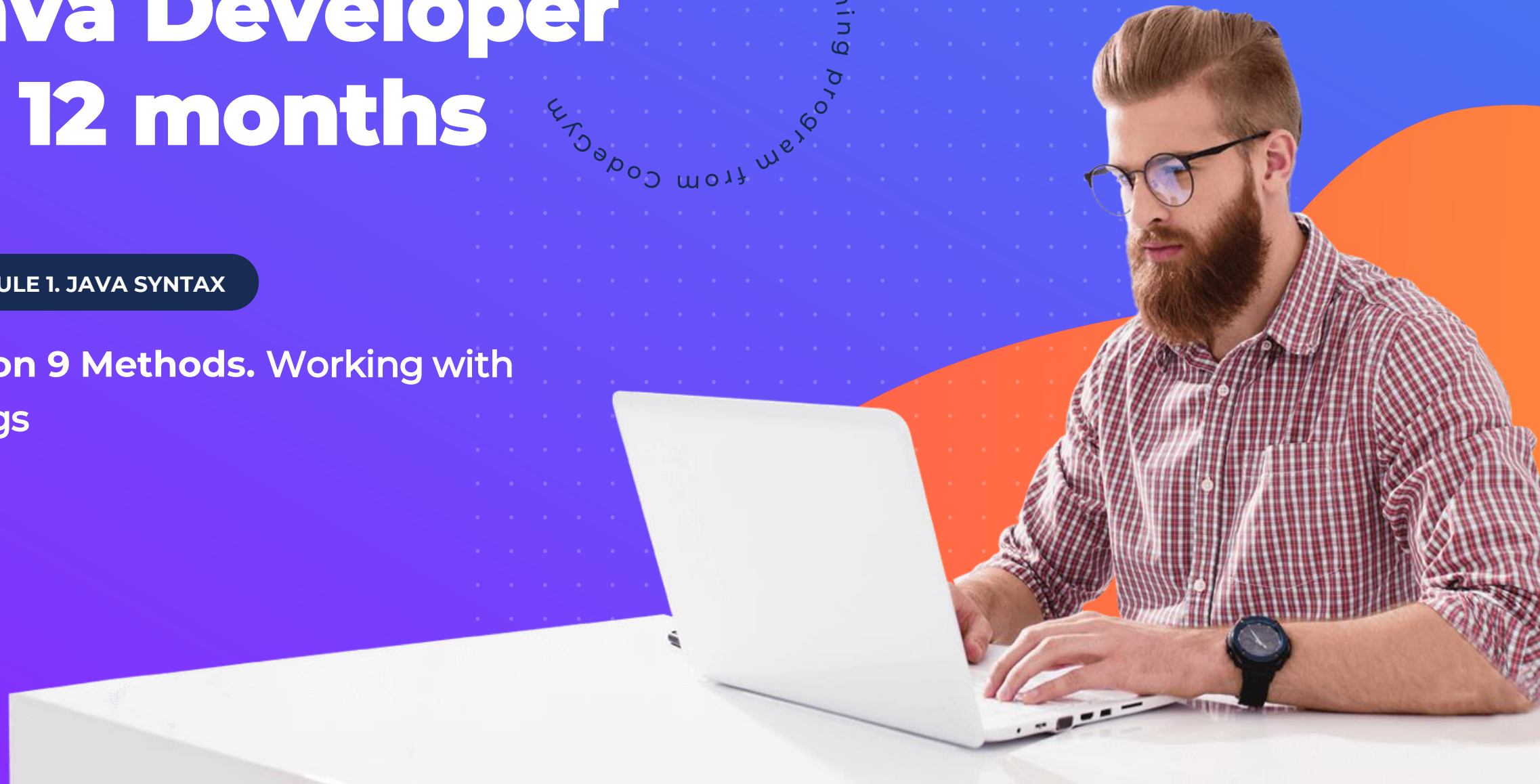
CODEGYM

Mentor-supported training program from CodeGym

Java Developer in 12 months

MODULE 1. JAVA SYNTAX

Lesson 9 Methods. Working with strings



CODEGYM

Lesson plan

- Constants
- Hiding variables (shadowing)
- Escaping characters
- Basic methods of the String class

CODEGYM

Constants

Constants are variables whose values cannot be changed. Creating an immutable variable looks the same as creating an ordinary one. The only difference is that before the variable's type you need to write the word **final**, like this:

```
final Type name = value;
```

If you assign a different value to a final variable, then your program simply won't compile.

Global constants

If you decide to declare global constants in your program, then you need to create *static class variables*, and make them **public** and **final**.

```
class Solution {
  public static final String SOURCE_ROOT = "c:\\projects\\my\\";
  public static final int DISPLAY_WIDTH = 1024;
  public static final int DISPLAY_HEIGHT = 768;
}
```

CODEGYM

Variable visibility

A property that determines whether you can use a variable in various parts of a class. It depends on where the variable is declared.

```
public class Solution {
   public int count = 0;
   public int sum = 0;

   public void add(int data) {
      sum = sum + data;
      int sum = data * 2;
      count++;
   }
}
```

```
count
count, sum
count, sum
count, sum, data
count, sum, data
count, sum, data
count, sum, data
count, sum
```

Visibility
starts from the moment of declaration — You cannot use a variable before it is declared.

CODEGYM

Variable shadowing

In the add method, we declared a local variable named sum. Until the end of the method, it shadows (or masks) the sum instance variable.

If an instance variable is shadowed by a local variable, the instance variable can be accessed using the this keyword.

```
public class Solution {
    public int count = 0;
    public int sum = 0;

    public void add(int data) {
        int sum = data * 2;
        this.sum = this.sum + data;
        count++;
    }
}
```

```
count
count, sum
count, sum
count, sum, data
count, this.sum, sum, data
count, this.sum, sum, data
count, this.sum, sum, data
count, sum
count, sum
```

The count and sum variables are available everywhere with or without the this keyword. On lines where the sum local variable shadows the sum instance variable, the sum instance variable can only be accessed using the this keyword.

CODEGYM

Escaping characters

Escaping is done using a special character.

Ordinarily, we call it a "backslash".
In Java, it is called a control sequence when combined with the character to be escaped.

There are a total of 8 special combinations like this, which are also called escape sequences.

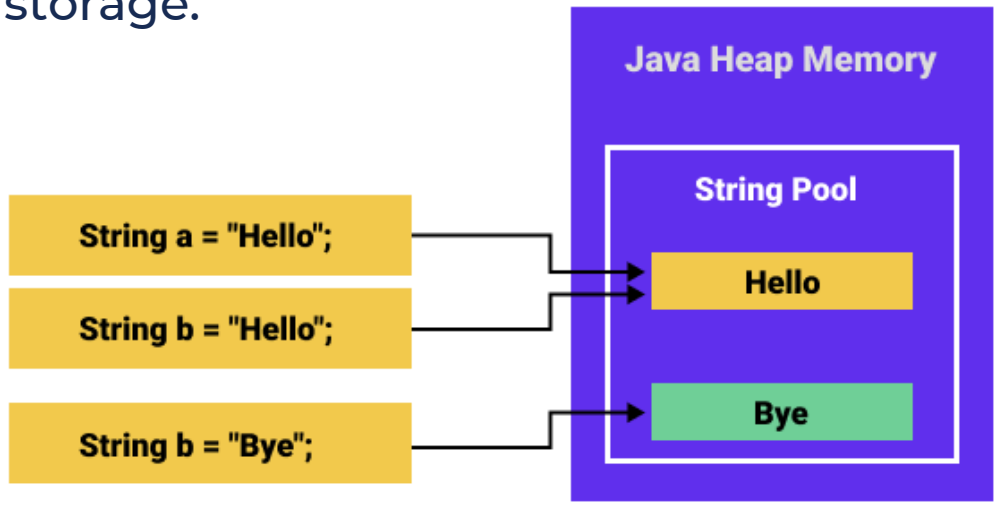
Code	Description
\t	Insert a tab character
\b	Insert a backspace character
\n	Insert a newline character
\r	Insert a carriage return character
\f	Insert a page feed character
\'	Insert a single quotation mark
\"	Insert a double quotation mark
\\	Insert a backslash

CODEGYM

String Pool

Every string specified in code as a string literal is stored in an area of memory called the StringPool while the program is running.

StringPool is a special array for storing strings. Its purpose is to optimize string storage.



intern() method

And the best part is that you can programmatically add any string to the **StringPool**. To do this, you just need to call the String variable's intern() method.

The intern() method will add the string to the **StringPool** if it is not already there and will return a reference to the string in the **StringPool**.

CODEGYM

Methods of the String class

The String class has a lot of methods. Here are a few:

Methods	Description
int length()	Returns the number of characters in the string
boolean isEmpty()	Checks whether the string is an empty string
boolean isBlank()	Checks that the string contains only whitespace characters: space, tab, new line, etc.
char charAt(int index)	Returns the character at the index position in the string.
char[] toCharArray()	Returns an array of the characters (a copy) that make up the string
byte[] getBytes()	Converts a string to a set of bytes and returns the array of bytes.
String[] split(String regex)	Splits a string into multiple substrings.
String join(CharSequence delimiter, elements)	Joins multiple substrings together
String intern()	Puts a string into the string pool.

CODEGYM

Comparing strings

One of the most common operations with strings is comparison. The String class has over ten different methods that are used to compare one string with another string.

Methods	Description
`boolean equals(String str)`	Strings are considered equal if all of their characters match.
`boolean equalsIgnoreCase(String str)`	Compares strings, ignoring the case of letters (ignores whether they are uppercase or lowercase)
`int compareTo(String str)`	Compares strings lexicographically. Returns 0 if the strings are equal. The return value is less than zero if the current string is less than the string parameter. The return value is greater than if the current string is greater than the string parameter.
`int compareToIgnoreCase(String str)`	Compares strings lexicographically while ignoring case. Returns 0 if the strings are equal. The return value is less than zero if the current string is less than the string parameter. The return value is greater than if the current string is greater than the string parameter.
`boolean regionMatches(int toffset, String str, int offset, int len)`	Compares parts of strings
`boolean startsWith(String prefix)`	Checks whether the current string starts with the string prefix
`boolean endsWith(String suffix)`	Checks whether the current string ends with the string suffix

CODEGYM

Searching for substrings

This is the second most popular operation after comparing strings.

Methods	Description
`int indexOf(String str)`	Searches for the string str in the current string. Returns the index of the first character of the first occurrence.
`int indexOf(String str, int index)`	Searches for the string str in the current string, skipping the first index characters. Returns the index of the occurrence.
`int lastIndexOf(String str)`	Searches for the string str in the current string, starting from the end. Returns the index of the first occurrence.
`int lastIndexOf(String str, int index)`	Searches for the string str in the current string from the end, skipping the first index characters.
`boolean matches(String regex)`	Checks whether the current string matches a pattern specified by a regular expression.

CODEGYM

Creating substrings

List of 8 methods that return substrings from the current string:

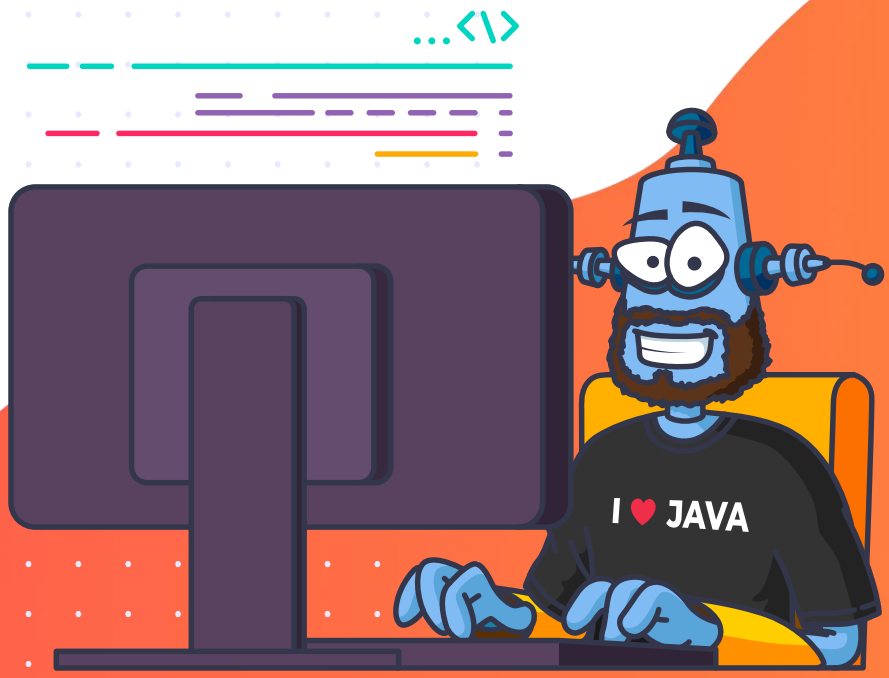
Methods	Description
String substring(int beginIndex, int endIndex)	Returns the substring specified by the index range beginIndex..endIndex.
String repeat(int count)	Repeats the current string n times
String replace(char oldChar, char newChar)	Returns a new string: replaces the character oldChar with the character newChar
String replaceFirst(String regex, String replacement)	Replaces the first substring, specified by a regular expression, in the current string.
String replaceAll(String regex, String replacement)	Replaces all substrings in the current string that match the regular expression.
String toLowerCase()	Converts the string to lowercase
String toUpperCase()	Converts the string to uppercase
String trim()	Removes all spaces at the beginning and end of a string

CODEGYM

Homework

MODULE 1. JAVA SYNTAX

Complete Level 10



CODEGYM

Answers to questions

MODULE 1. JAVA SYNTAX