CODEGYM

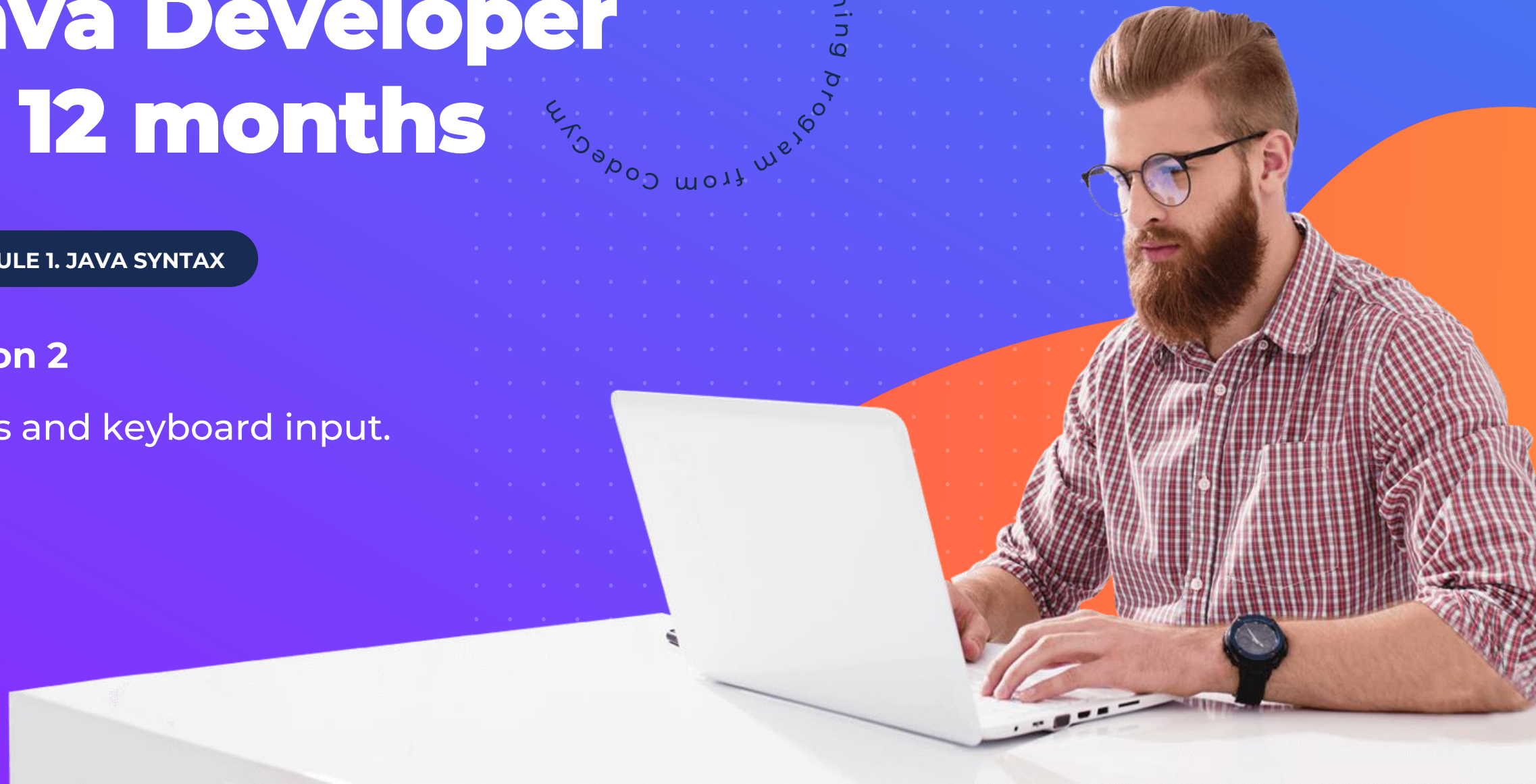
Java Developer in 12 months

Mentor-supported training program from CodeGym

MODULE 1. JAVA SYNTAX

Lesson 2

Types and keyboard input.



CODEGYM

Lesson plan

- String variables
- Concatenation
- storing primitive variables in memory
- storing String variables in memory
- System.in, Scanner
- methods of the Scanner class

CODEGYM

String type: creating a variable

`String` is a type for storing strings (text).

```
String name;
```

Example

`String name;`	A string variable named `name` is created
`String message;`	A string variable named `message` is created
`String text;`	A string variable named `text` is created

All objects in Java can be converted to `String`

CODEGYM

Assigning values to String variables

```
name = " value ";
```

statement for assigning a value to a `String` variable

Example

`String name = "Steve";`	The `name` variable contains the text `Steve`
`String city = "New York";`	The `city` variable contains the text `New York`
`String message = "Hello!";`	The `message` variable contains the text `Hello!`

```
String name1 = " value1 ", name2 = " value2 ", name3 = " value3 ";
```

Initializing `String` variables

CODEGYM

What is concatenation?

Concatenation is merging or joining strings:

" value1 " + " value2 "

Example

`String name = "Steve" + "Steve";`	`name` contains the string `SteveSteve`
`String city = "New York" + "Steve";`	`city` contains the string `New YorkSteve`
`String message = "Hello! " + "Steve";`	`message` contains the string `Hello! Steve`

To indicate one or more spaces, you need to write them in code and then wrap them in double quotes.

CODEGYM

Converting a string to a number

To convert a string to a number, we use a special method of the `Integer` class:

Example

`String str = "123";` `int number = Integer.parseInt(str);`	`number` contains the number `123`;
`int number = Integer.parseInt("321");`	`number` contains the number `321`
`int number = Integer.parseInt("321" + 0);`	`number` contains the number `3210`
`int number = "321";`	This won't compile: the variable is an `int`, but the value is a `String`

CODEGYM

Converting an object/primitive to a string

To convert an instance of any Java class or any primitive data type to a string, you can use the String.valueOf() method:

```
public class Solution {

    public static void main(String args[]){
        double d = 102939939.939;
        boolean b = true;
        long l = 1232874;
        char[] arr = { 'a', 'b', 'c', 'd', 'e', 'f', 'g' };

        System.out.println("Return value: " + String.valueOf(d));
        System.out.println("Return value: " + String.valueOf(b));
        System.out.println("Return value: " + String.valueOf(l));
        System.out.println("Return value: " + String.valueOf(arr));
    }
}
```

```
Return value: 1.029399399939E8
Return value: true
Return value: 1232874
Return value: abcdefg
```

CODEGYM

Converting to a String

When we add a `String` to another type, it is converted to a `String`:

Example

`int a = 5;` `String name = "Steve" + a;`	`name` contains the string `Steve5`
`int a = 5;` `String city = a + "New York" + a;`	`city` contains the string `5New York5`

`int number = 10;`
`String code = "Yo";`
`String message = "Hello! " + number + code;`
	`message` contains the string `Hello! 10Yo`

You can't perform arithmetic operations with the `String` type. Even if the entire string consists of digits.

CODEGYM

Methods for working with Strings

The `length()` method lets you get the length of a string, i.e. how many characters it contains.

```
String name = "Rome";
int count = name.length();
```

`count` contains the value `4`

The `toLowerCase()` method converts all characters in a string to lowercase:

```
String name = "Rom";
String name2 = name.toLowerCase();
```

`name2` contains the string `rom`

The `toUpperCase()` method converts all characters in a string to uppercase:

```
String name = "Rom";
String name2 = name.toUpperCase();
```

`name2` contains the string `ROM`

CODEGYM

What is main memory?

Every program is loaded into **main memory** before being executed.
The main memory contains the **program code** and the **program data**.

	A	B	C	D	E	F	G	H	I
1									
2		13	11						
3									
4	A	m	i	g	o				
5		G	e	r	m	e	s		
6									
7									

The program and program data are stored in **memory** when the program is running.

All computer memory is comprised of small cells called **bytes**.

Each cell has a **unique identifier**, or number, associated with it: 0, 1, 2, 3, ...;

CODEGYM

How variables are stored in memory

Java has 4 data types for storing integers: `byte`, `short`, `int` and `long`:

Type	Size in bytes	Origin of the type's name
`byte`	1	`Byte` is a deliberate respelling of bite to avoid confusion with bit
`short`	2	Short for `Short Integer`
`int`	4	Short for `Integer`
`long`	8	Short for `Long Integer`

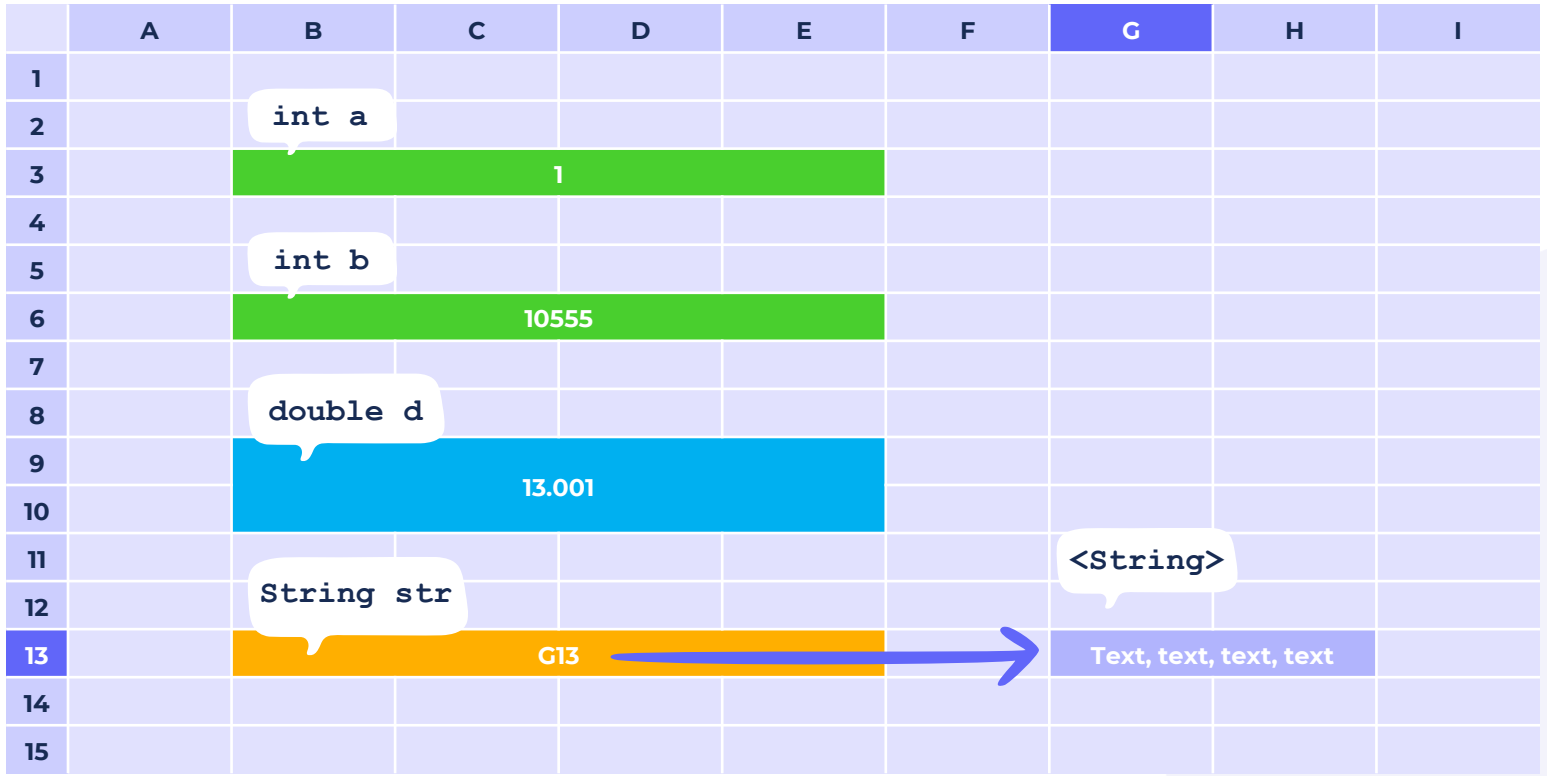
And 2 real types — `float` and `double`:

Type	Size in bytes	Origin of the name
`float`	4	Short for `Floating Point Number`
`double`	8	Short for `Double Float`

The address of a variable is the address of the first cell of the allocated memory block.

CODEGYM

Features of declaring a String array in memory



The data of a **`String`** object (the text) is placed in a special allocated block of memory

The **`String`** object is stored in a separate block of memory. The address of its first cell is stored in the **`str`** variable.

CODEGYM

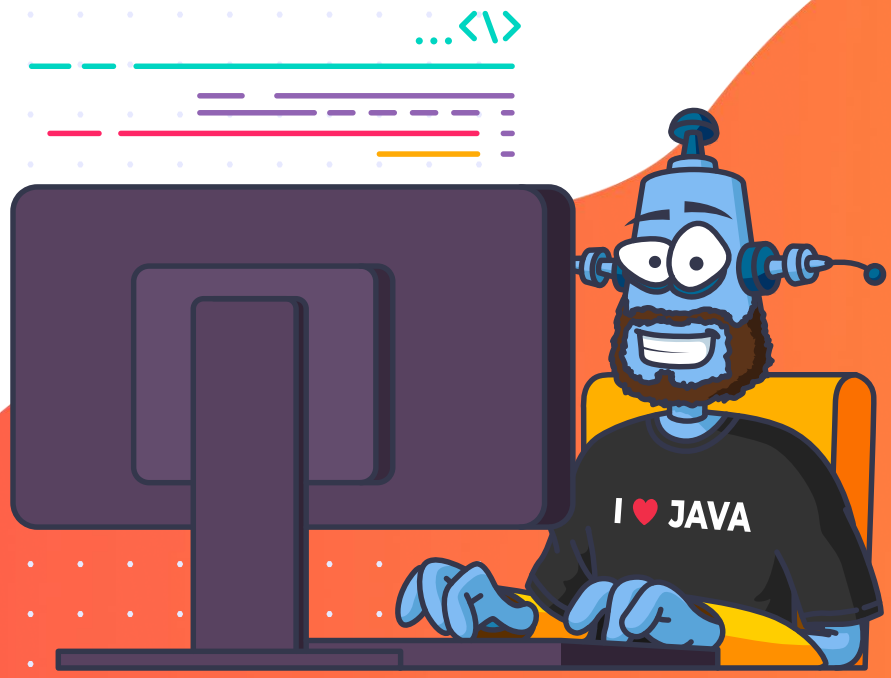
Primitive types in Java

The table contains information about the size, range of possible values, and default values of primitive types:

Type	Size in bytes	Size in bits	Possible values (from..to)	Default value
boolean	—	1	true or false	false
byte	1	8	-128..127	0
short	2	16	-32,768..32,767	0
int	4	32	-2,147,483,648..2,147,483,647	0
long	8	64	-9,223,372,036,854,775,808..9,223,372,036,854,775,807	0
char	2	16	0..65,535	'\u0000'
float	4	32	-3.4E+38..3.4E+38 (IEEE Standard 754)	0.0
double	8	64	-1.7E+308..1.7E+308 (IEEE Standard 754)	0.0

CODEGYM

WORKING WITH INPUT



MODULE 1. JAVA SYNTAX

CODEGYM

Reading from the console, System.in

`System.in` is a special object for data input

It lets us read data from the keyboard one character at a time.

CODEGYM

Scanner class

The `Scanner` class (full name: `java.util.Scanner`) can read data from different sources, e.g. the console, files, and the Internet.

Example

```
Scanner console = new Scanner(System.in); String name =
console.nextLine();
int age = console.nextInt();
```

Create a `Scanner` object.
Read a line of text from the keyboard.
Read a number from the keyboard.

CODEGYM

Creating Scanner objects

```
Scanner console = new Scanner(System.in);
```

declaration of a `Scanner` variable named `console`

Create a new object (`new` keyword) whose type is `Scanner`,
passing in the `System.in` object as the data source for the newly created `Scanner` object

CODEGYM

List of methods

Here's the general appearance of this statement:

```
variable . method ( parameters );
```

Example

```
System.out.println("Hello");
System.out.println(1);
```

If you don't need to pass arguments
to the method, then use empty parentheses:

```
variable . method ();
```

Example

```
System.out.println();
```

CODEGYM

Console input

Statement for reading a string from the keyboard:

```
String str = console.nextLine();
```

Statement for reading a number from the keyboard:

```
int number = console.nextInt();
```

Statement for reading a fractional number from the keyboard:

```
double number = console.nextDouble();
```

CODEGYM

Reading data from a String

General form:

```
String str = " text ";

Scanner scanner = new Scanner (str);
```

Example

```
import java.util.Scanner;
public class Solution {
  public static void main(String[] args)
  {
   String str = "10 20 40 60";
   Scanner scanner = new Scanner(str);
   int a = scanner.nextInt();
   int b = scanner.nextInt();

   System.out.println(a + b);
  }
}
```

```
// a == 10;
// b == 20;
The screen output will be:
30
```

CODEGYM

Homework

MODULE 1. JAVA SYNTAX

Complete Level 3

CODEGYM

Answers to questions

MODULE 1. JAVA SYNTAX