CODEGYM

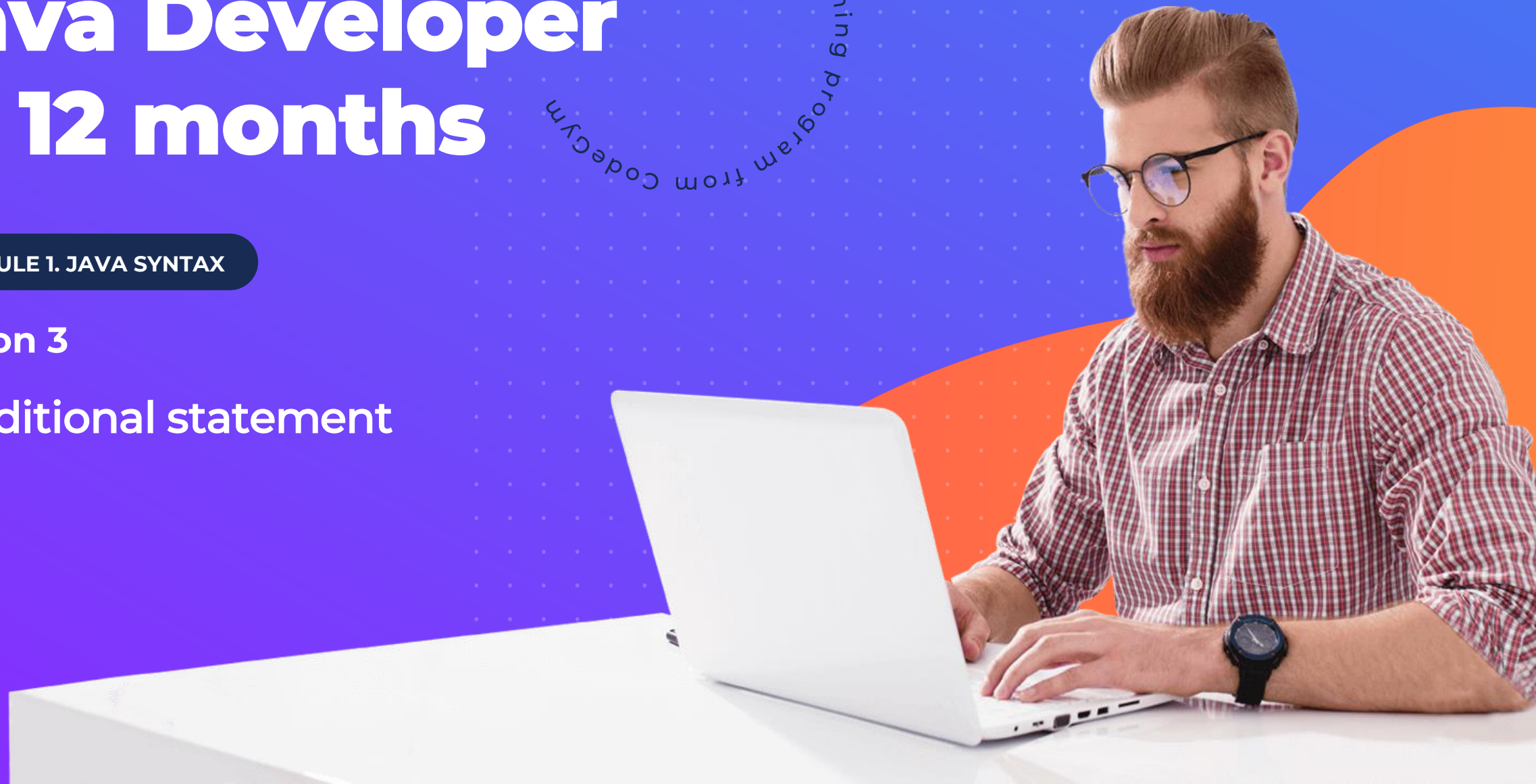
Mentor-supported training program from CodeGym

Java Developer in 12 months

MODULE 1. JAVA SYNTAX

Lesson 3

Conditional statement



CODEGYM

Lesson plan

- Statement & expression
- The if conditional statement, if-else, if-else-if
- Blocks of statements
- Nested blocks of statements (nested ifs)
- boolean type
- Comparison operators
- Logical AND, OR, NOT
- Ternary operator
- Comparing primitives and String

CODEGYM

Expressions and statements

In Java, it is helpful to distinguish between two categories:

Statements

Expressions

A statement is usually said to be executed, while an expression is said to be calculated. But that's not the most important thing.

The main difference between a statement and an expression is that evaluating an expression has a result.

And this result has a type, and it can be assigned to a variable or used in some other expression.

Code	Notes
int x;	Statement
(a < 10)	Expression whose type is boolean
i++;	Expression whose type is the same as the type of the i variable
x = 5;	Expression whose type is the same as the type of the x variable

CODEGYM

What is the boolean type for?

The primitive boolean type get its name from George Boole, who first defined a branch of algebra that uses only two values — true and false. It is impossible to assign any other values to boolean variables.

Code	Explanation
`boolean isOK = true;`	The boolean isOK variable contains the value true
`boolean hasError = false;`	The boolean hasError variable contains the value false
`int age = 70;` `boolean isSenior = (age > 65);`	The boolean isSenior variable contains the value true

`int record = 612;`
`int value = 615;`
`boolean hasNewRecord = (value > record);`
	The boolean hasNewRecord variable contains the value true

CODEGYM

Comparison operators

In Java, as in other programming languages, it is often necessary to compare variables with one another. And Java has just the operators you need to make comparisons:

Operator	Explanation	Example
<	Less than	a < 10
>	Greater than	b > a
<=	Less than or equal	a <= 10
>=	Greater than or equal	speed >= max
==	Equal to	age == 18
!=	Not equal to	time != 0

CODEGYM

Conditional statement

MODULE 1. JAVA SYNTAX

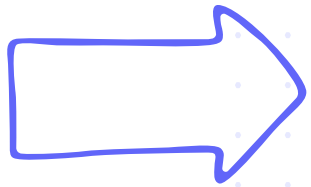
CODEGYM

The if-else statement

An if-else conditional statement is a special construct you can use to execute different blocks of statements depending on the truth value of a condition

```
if (condition)
    statement1;
else
    statement2;
```

The if-else conditional operator



```
If the condition is true, then
    execute statement1;

Otherwise
    execute statement2;
```

The if-else conditional operator in plain language

CODEGYM

Example of an if-else

If the condition is true, then statement 1 is executed, otherwise statement 2 is executed. Never are both statements executed.

Code	Explanation

`int age = 17;`
`if(age< 18) {`
`  System.out.println("You are still a child");`
`  System.out.println("Don't talk back to adults");`
`}`
 	
`else {`
`  System.out.println("You are now an adult");`
`  System.out.println("And thus ends your youth");`
`}`
	The screen output will be:
	
	You are still a child
	Don't talk back to adults

CODEGYM

Block of statements

If the condition is satisfied (or not) and you want your program to execute several commands, you can combine them into a block of statements.

To execute several statements, combine them in a block in curly braces:

```
{
statement1;
statement2;
statement3;
}
```

CODEGYM

Examples of an if-else statement combined with a block of statements:

Code	Explanation

`int temperature = 5;`
 	
`if(temperature< 0) {`
`  System.out.println("It's freezing outside");`
`  System.out.println("Put on a hat");`
`}`
`else System.out.println("It's warm");`
	The screen output will be:
	It's warm

`int age = 21;`
`if (age == 18) {`
`  System.out.println("You've been drafted for military service");`
`}`
`else {`
`}`
	The empty block will be executed.
	Nothing will be displayed on the screen

CODEGYM

Abbreviated form of the if statement

If statements(s) need to be executed only if the condition is true and there are no commands to be executed when the condition is false, then the if statement is used, which is concise and omits the else block.

```
if (condition)
  statement1;
```

For example, we can specify this command: If Bus No. 62 has arrived, then get aboard, but don't react if the bus isn't here.

```
int bus = 62;
if (bus == 62) {
   System.out.println(”Get aboard");
}
```

The screen output will be:
Get aboard

CODEGYM

Sequence of if statements

`If-else` statements can be nested within one another, which makes it possible to implement complex logic in a program

Multiple conditions:

- If the temperature is greater than 20 degrees, then put on a shirt
- If the temperature is greater than 10 degrees and less than (or equal to) 20, then put on a sweater
- If the temperature is greater than 0 degrees and less than (or equal to) 10, then put on a raincoat
- If the temperature is less than 0 degrees, then put on a coat.

```
int temperature = 9;

if (temperature > 20)
   System.out.println("put on a shirt");
else // Here the temperature is less than (or equal to) 20
{
   if (temperature > 10)
      System.out.println("put on a sweater");
   else // Here the temperature is less than (or equal to) 10
   {
      if (temperature > 0)
         System.out.println("put on a raincoat");
      else // Here the temperature is less than 0
         System.out.println("put on a coat");
   }
}
```

CODEGYM

Using boolean variables

You can use a boolean variable in the condition of an if statement:

Code	Equivalent

`int age = 70;`
 	
`boolean isSenior = (age > 65);`
`if (isSenior)`
`  System.out.println("Time to retire");`
	`int age = 70;`
	
	`if (age > 65)`
	`  System.out.println("Time to retire");`

CODEGYM

Logical operators

There are three logical operators in Java: AND (&&), OR (||) and NOT (!)

Logical operator	Expectation	Reality
AND (∧)	and	&&
OR (∨)	or	||
NOT (¬)	not	!

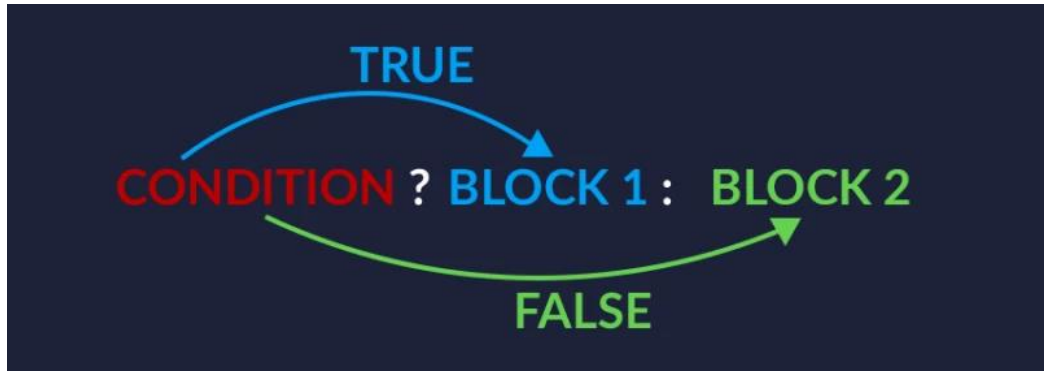
Here are some examples of using logical operators in Java:

Expression	Interpretation	Explanation
(0 < a) && (a < 100)	(0 < a) and (a < 100)	(0 < a) AND (a < 100)
(!a) && (!b)	(not a) and (not b)	(NOT a) AND (NOT b)
!(!a || !b)	not((not a) or (not b))	NOT((NOT a) OR (NOT b))

CODEGYM

Ternary operator

"Ternary" means "composed of three parts". This is an alternative to the `if-else` conditional statement.



If condition is true, then Expression 1 is evaluated, otherwise Expression 2 is evaluated.

Example. Calculate the minimum of two numbers:

With if-else

```
int age = 25;
int money;
if (age > 30)
   money = 100;
else
   money = 50;
```

With ternary operator

```
int age = 25;
int money = age > 30 ? 100 : 50;
```

CODEGYM

Comparing strings

There are two operators you can use to compare string variables:

`==` (equal to)

`!=` (not equal to)

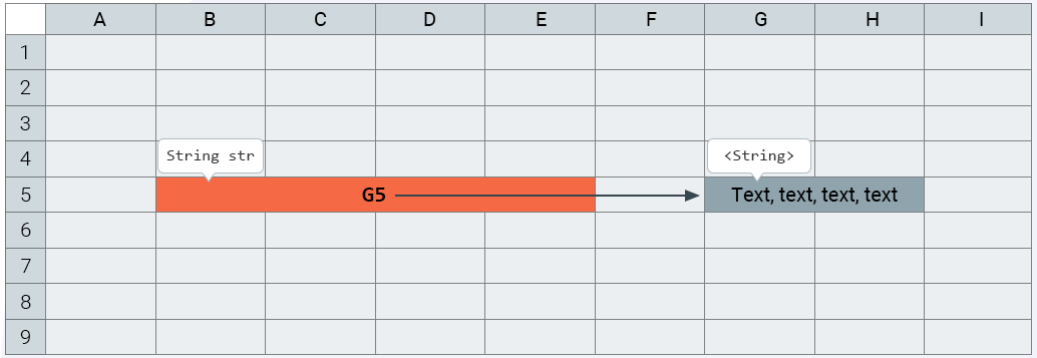
Example:

```
String text = "Hello";
String message = text;
//The addresses are equal
System.out.println(text == message);
```

The message and text variables refer to (store the address of) the same object.

You can't use the "greater than", "less than", or "greater than or equal to" operators — the compiler won't allow it.

Two blocks of memory are used to store strings: one block stores the text itself while the second block stores the address of the first block.



CODEGYM

Comparing strings by content

To help us with this, Java's String class has the **equals** method

```
string1.equals(string2)
```

This method returns true if the strings are the same, and false if they are not the same.

Code:	Note

String s1 = "Hello";
String s2 = "HELLO";
String s3 = s1.toUpperCase();
 	
System.*out*.println(s1.equals(s2));
System.*out*.println(s1.equals(s3));
System.*out*.println(s2.equals(s3));
	// Hello
	// HELLO
	// HELLO
	
	false // They are different
	false // They are different
	true // They are the same, even though the addresses are different

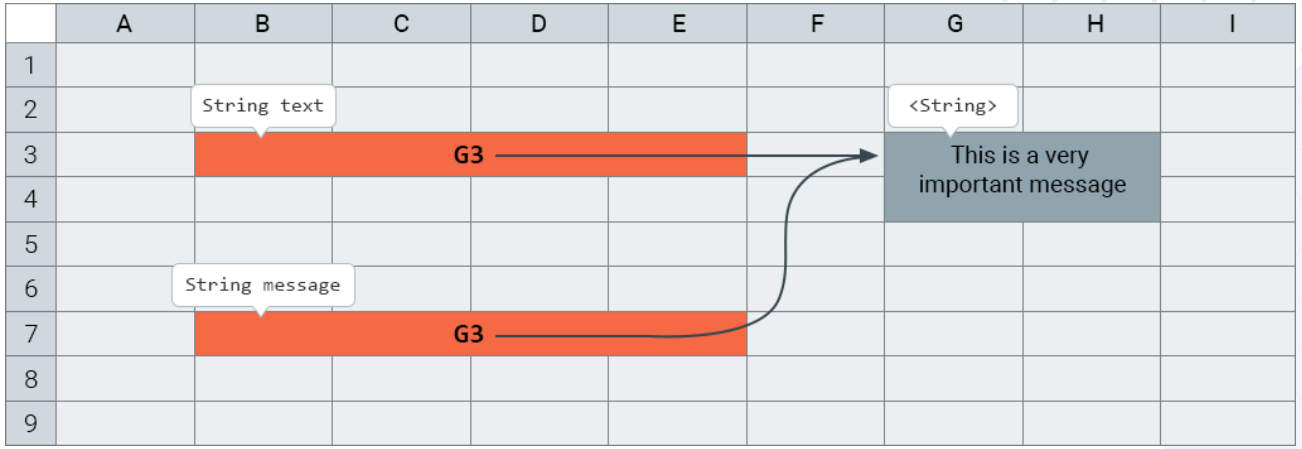
CODEGYM

An interesting nuance of string comparison

If the Java compiler finds multiple identical strings in your code (specifically in your code), then it will create only a single object for them in order to save memory.

```
String text = "This is a very important message"; String message = "This is a very important message";
```

And here's what memory will contain as a result:



CODEGYM

Homework

MODULE 1. JAVA SYNTAX

Complete Level 4

CODEGYM

Answers to questions

MODULE 1. JAVA SYNTAX