НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Algorithms & Programming

(p.1 - basics)

Yevhen Berkunskyi, NUoS
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Computers – for what?

A computer is a universal machine – one machine that serves many purposes.

Computers:

- Make calculations
- Can store a huge amount of information
- allow the exchange of information, regardless of location …

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

What is programming?

Programming today – is a race between software engineers striving to build bigger and better idiot-proof programs, and the Universe trying to produce bigger and better idiots.

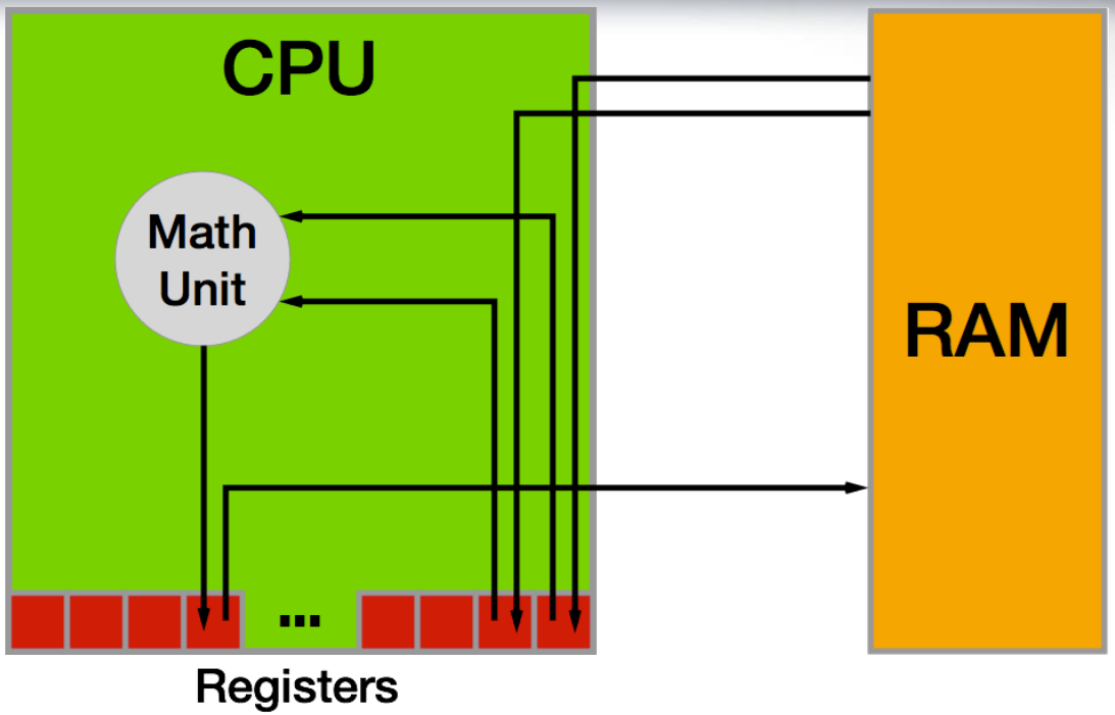
So far, the Universe is winning

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

How a computer works

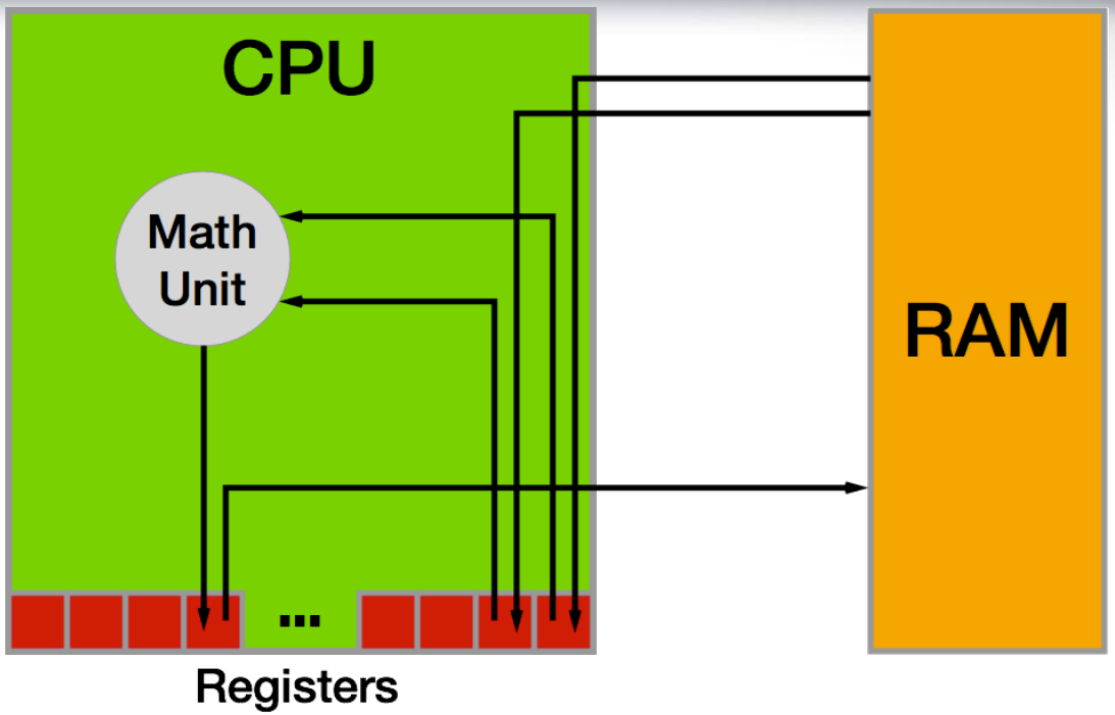
- Did you know that computer is not very smart on its own ?
- The power of computers is all derived from how they’re programmed by people
- If you want to successfully harness the power of a computer…
- … it’s important to understand how computers work

How a computer works



- The CPU stores the numbers it acts upon in small memory units called **registers**.
- The CPU is able to read numbers into registers from the computer's main memory, known as **Random Access Memory** (**RAM**)

How a computer works



- The CPU stores the numbers it acts upon in small memory units called **registers**.
- The CPU is able to read numbers into registers from the computer's main memory, known as **Random Access Memory** (**RAM**)
- It's also able to write the number stored in a register back into RAM.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

How computer works

- Each time the CPU makes an addition, a subtraction, a read from RAM or a write to RAM, it's executing a single **instruction**.
- Each computer program is usually made up of thousands to millions of instructions.
- A complex computer program such as your operating system, be it iOS, Android, macOS, Windows or Linux (yes, they're computer programs too!), may have many millions of instructions in total.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Programming languages

- It’s entirely possible to write individual instructions to tell a computer what to do, but for all but the simplest programs, it would be immensely time-consuming and tedious.
- This is because most computer programs aim to do much more than simple math — computer programs let you surf the internet, manipulate images, and allow you to chat with your friends.

Programming languages

- Instead of writing individual instructions, you write **code** in a specific **programming language**, which in our case will be Kotlin.
- This code is put through a computer program called a **compiler**, which converts the code into instructions the CPU knows how to execute.
- Each line of code you write will turn into many instructions — some lines could end up being tens of instructions!

Compiling

- In the case of Kotlin, with its origins as a language on the **Java Virtual Machine** or **JVM**, there is an extra layer between the compiler and the OS.
- The Kotlin compiler creates what is known as **bytecode**, which gets run on the JVM and converted to native code along the way. Kotlin began on the JVM but now it is possible to compile Kotlin directly to native code

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

About Kotlin

- Kotlin – is new programming language
  - Kotlin v1.0 was released on 15 February 2016. The name comes from Kotlin Island, near St. Petersburg.
- Kotlin – is compiling language
  - Kotlin is compiling to a Java-bytecode, that makes it cross-platform
- Kotlin – is statically typed language
- Kotlin realized many paradigms:
  - Kotlin realizes procedural, generified, objective-oriented, functional and other paradigms…
- Kotlin works on top of JVM
  - In Kotlin you can use Java libraries and it works!

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

And we will learn that all stuff?

IDE

- In this course we'll create Kotlin projects is **IntelliJ IDEA** from JetBrains.
- JetBrains is also the company behind the Kotlin language itself, so Kotlin development is very tightly integrated into IntelliJ IDEA.

IntelliJ IDEA

- IntelliJ IDEA is an **Integrated Development Environment**, or **IDE**, and is similar to other IDEs such as **Visual Studio** and **Xcode**
- IntelliJ IDEA provides the foundation of many other IDEs from JetBrains, including **Android Studio** for Android app development, **PyCharm** for Python programming and **CLion** for C and C++ programming.
- You use an IDE to write code in an editor, **compile** the code into a form that can be run on your computer, see output from your program, fix issues in your code and much more!

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

IntelliJ IDEA

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Getting started with Kotlin

- The Kotlin compiler generates bytecode or executable code from your source code.
- To accomplish this, it uses a detailed set of rules you will learn about in this course
- Sometimes these details can obscure the big picture of *why* you wrote your code a certain way or even what problem you are solving.
- *What can we do with this?*

Comments

- Kotlin, like most other programming languages, allows you to document your code through the use of what are called **comments**.
- These allow you to write any text directly along side your code which is ignored by the compiler.
- The first way to write a comment is like so:

```
// This is a comment. It is not executed
```

This is a single line comment

Comments

You could stack these up like so to allow you to write paragraphs:

```
// This is also a comment.
// Over multiple lines.
```

However, there is a better way to write comments which span multiple lines. Like so:

```
/* This is also a comment.
Over many..
many...
many lines. */
```

This is a **multi-line comment**. The start is denoted by /* and the end is denoted by
*/. Simple!

Nested comments

- Kotlin also allows you to nest comments, like so:

```
/* This is a comment.
/* And inside it
is
another comment.
*/
Back to the first.
*/
```

This might not seem particularly interesting, but it may be if you have seen other programming languages. Many do not allow you to nest comments like this

Printing out

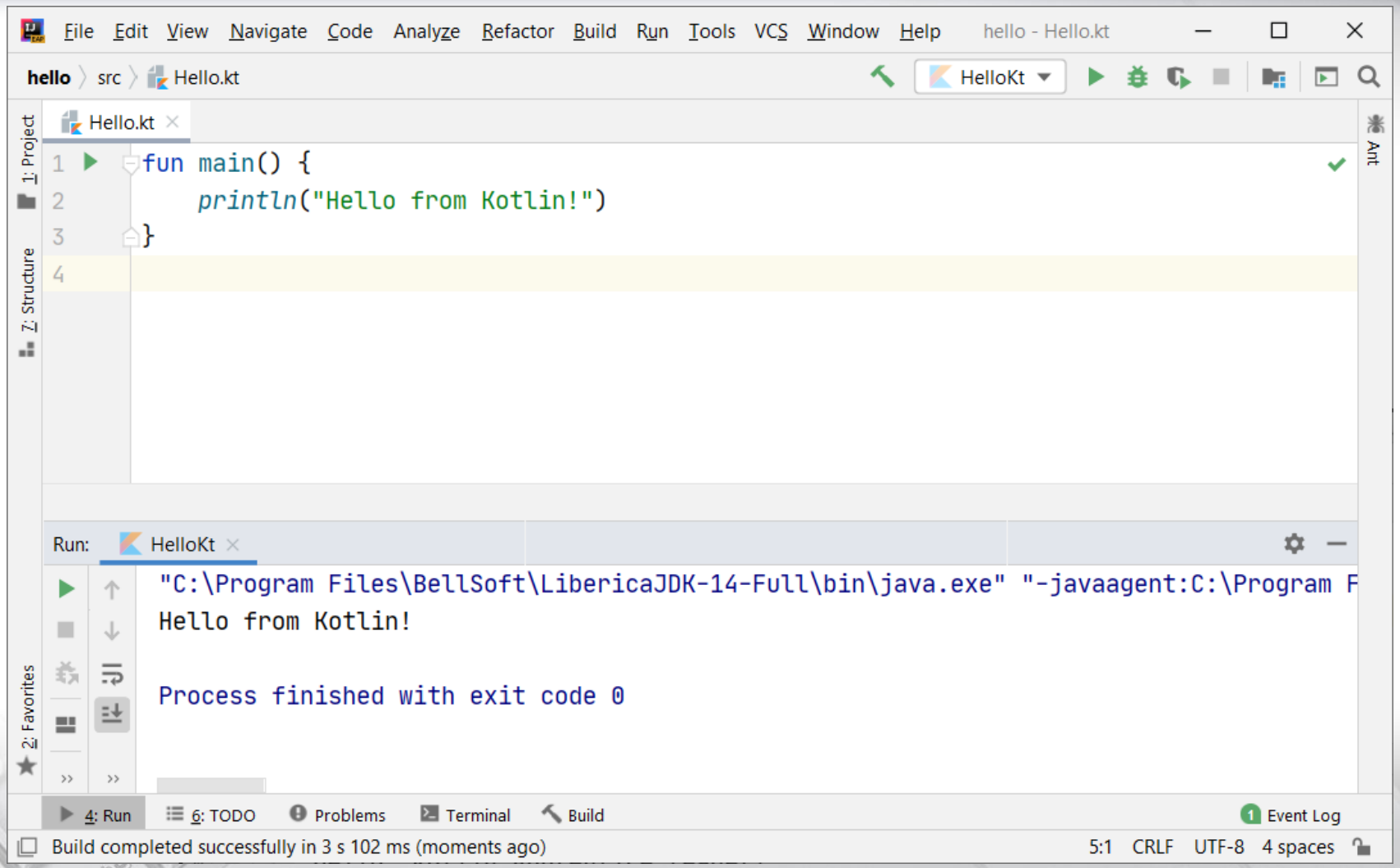
- It’s also useful to see the results of what your code is doing.
- In Kotlin, you can achieve this through the use of the println command.
- println will output whatever you want to the **console**.

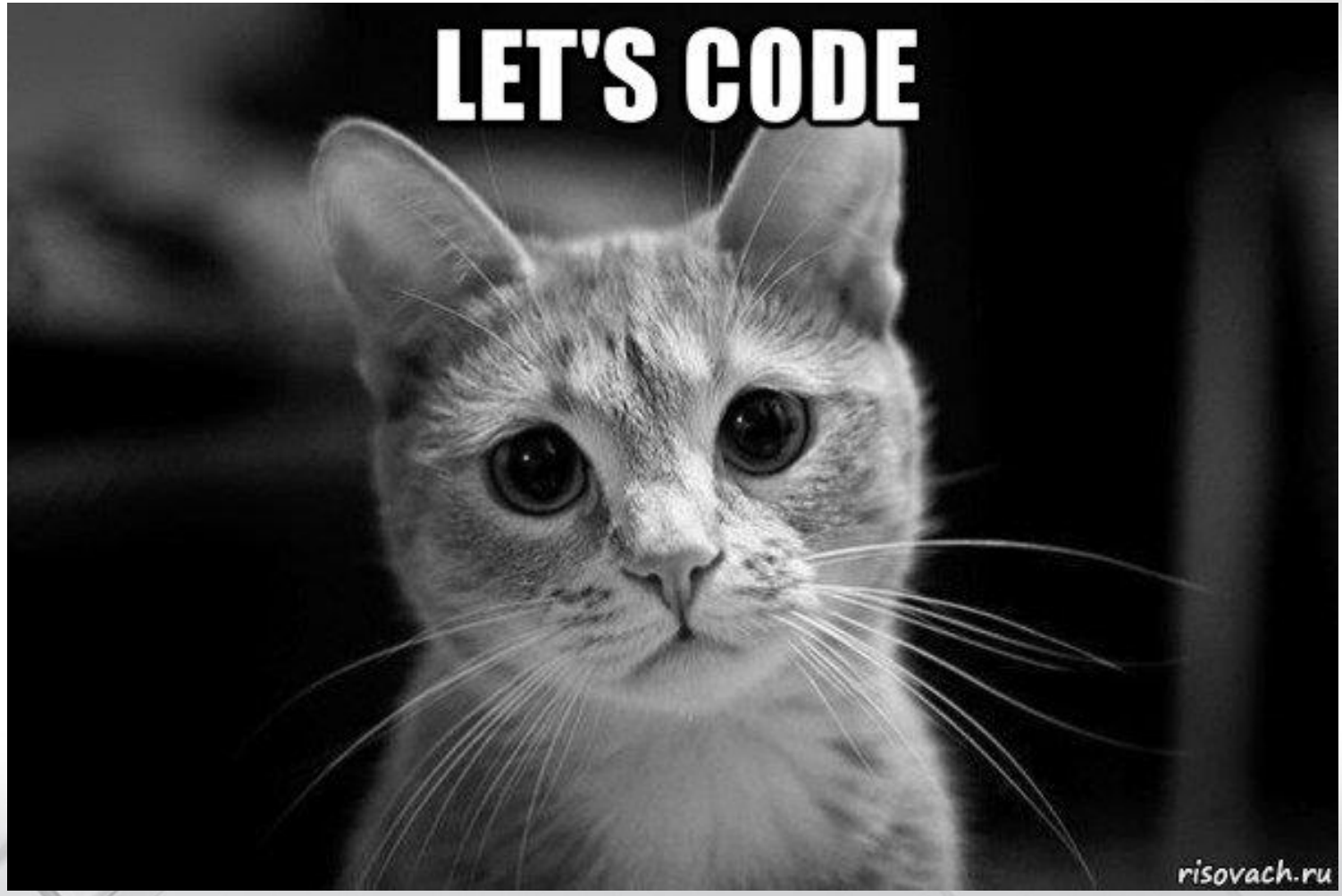
For example, consider the following code:

```
println("Hello from Kotlin!")
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Printing out




LET'S CODE
risovach.ru

Arithmetic operations

- When you take one or more pieces of data and turn them into another piece of data, this is known as an **operation**.
- All operations in Kotlin use a symbol known as the **operator** to denote the type of operation they perform.
- Consider the four arithmetic operations you learned in your early school days: addition, subtraction, multiplication and division.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Arithmetic operations

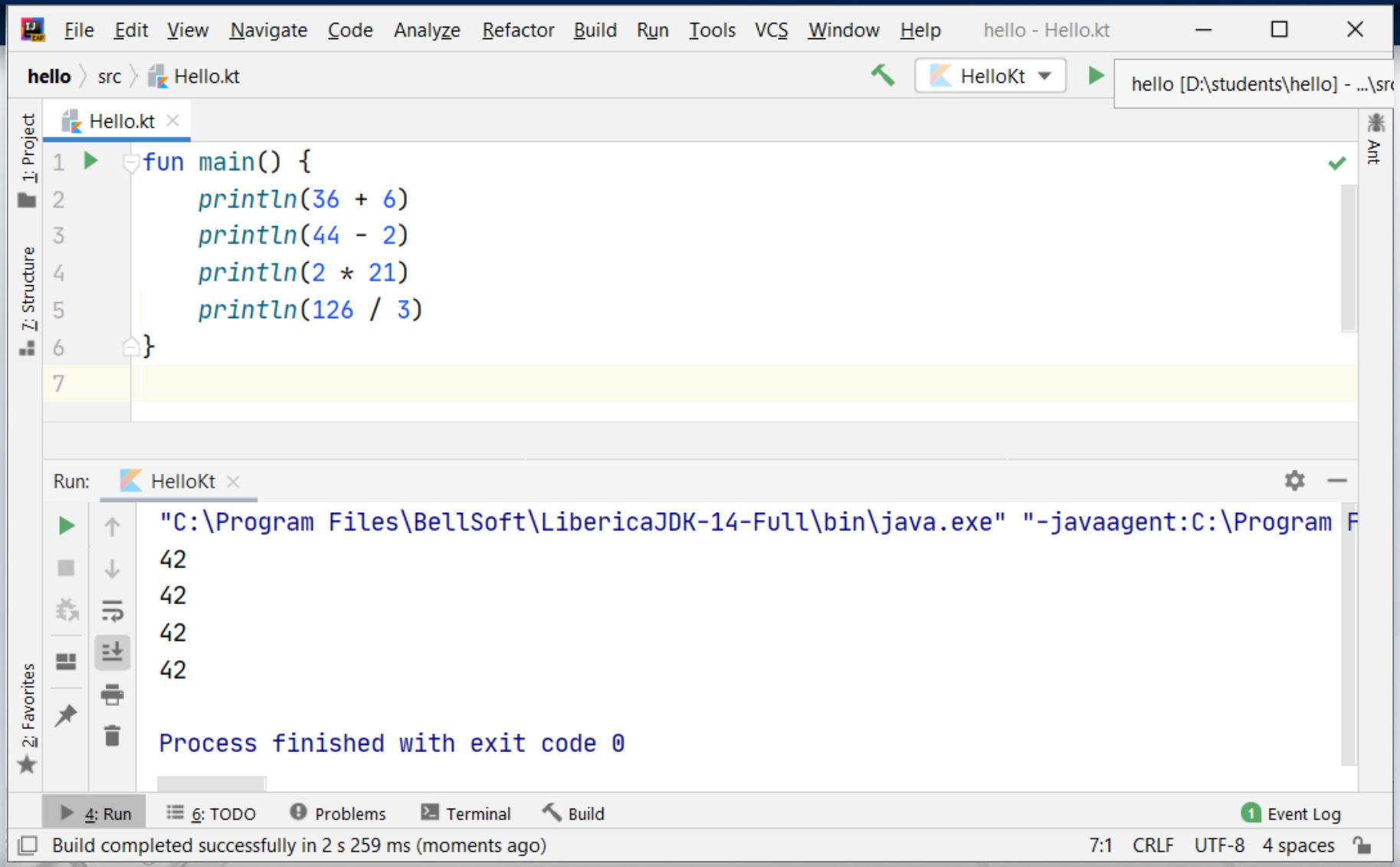
- For these simple operations, Kotlin uses the following operators:
  - Add: +
  - Subtract: -
  - Multiply: *
  - Divide: /

  These operators are used like so:

```
2 + 6
10 - 2
2 * 4
24 / 3
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Printing & Arithmetic



Decimal numbers

- All of the operations above have used whole numbers, more formally known as **integers**.
- However, as you know, not every number is whole.
- As an example, consider the following:

```
22 / 7
```

This, you may be surprised to know, results in the number 3. This is because if you only use integers in your expression, Kotlin makes the result an integer also.

```
22.0 / 7.0
```

This time, the result is 3.142857142857143 as expected

The remainder operation

- Kotlin also has more complex operations you can use, all of them standard mathematical operations, just less common ones.
- The first of these is the **remainder** operation, also called the modulo operation.
- In division, the denominator goes into the numerator a whole number of times, plus a remainder.
- In Kotlin, the remainder operator is the % symbol, and you use it like so:

```
28 % 10
```

In this case, the result equals 8, because 10 goes into 28 twice with a remainder of 8.

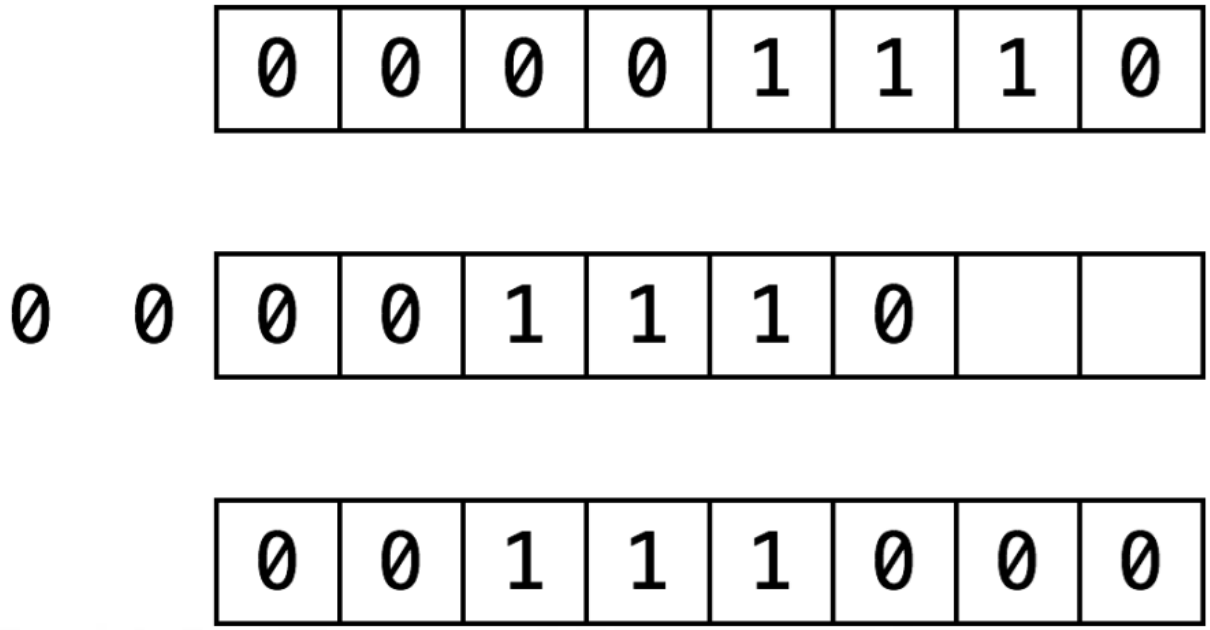
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Shift operations

- The **Shift left** and **Shift right** operations take the binary form of a decimal number and shift the digits left or right, respectively.
- Then they return the decimal form of the new binary number.
- For example, the decimal number 14 in binary, padded to 8 digits, is 00001110. Shifting this left by two places results in 00111000, which is 56 in decimal.

Shift operations

- Here's an illustration of what happens during this shift operation:



The digits that come in to fill the empty spots on the right become 0. The digits that fall off the end on the left are lost. Shifting right is the same, but the digits move to the right.

Shift operations

The Kotlin functions for these two operations are as follows:

- Shift left: shl
- Shift right: shr

These are **infix** functions that you place in between the operands so that the function call looks like an operation:

```
1 shl 3
32 shr 2
```

Order of operations

When you calculate a value, you'll want to use multiple operators:

```
(((12000 / (4 * 10)) - 32) shr (29 % 5)) + 26
```

Parentheses in Kotlin serve two purposes: to make it clear to anyone reading the code — including yourself — what you meant, and to disambiguate.
For example, consider the following:

```
20 + 350 / 5
```

What is the value of it?

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Order of operations (1/2)

Order	Operation	Associativity	Description
1	[ ]	left-to-right	Index operation
	( )		parentheses
	.		Accessing to class (object) member
2	++	left-to-right	Postfix increment
	– –		Postfix decrement
3	++	right-to-loft	Prefix increment
	– –		Prefix decrement
4	*	left-to-right	Multiplying
	/		Division
	%		Remainder
5	+	left-to-right	Addition
	–		Subtraction

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Order of operations (2/2)

Order	Operation	Associativity	Description
6	**shr**	left-to-right	Shift to right
	shl		Shift to left
7	**<**	left-to-right	Less than
	<=		Less or equals
	>		Greater than
	>=		Greater or equals
8	**==**	left-to-right	Equals
	!=		Not equals
9	**&&**	left-to-right	logical **AND**
10	**||**	left-to-right	logical **OR**
11	**=**	right-to-left	assignment
	***=**		Multiply and assign
	/=		Division and assign
	%=		Remainder and assign
	+=		Addition and assign
	-=		Subtraction and assign
13	**,**	left-to-right	comma

Math functions

- Kotlin also has a vast range of math functions in it's **standard library** for you to use when necessary.
- You never know when you need to pull out some trigonometry, especially when you're a pro at Kotlin and writing those complex games!

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Math functions

- For example, consider the following:

```
sin(45 * PI / 180)
// 0.7071067811865475
cos(135 * PI / 180)
// -0.7071067811865475
```

Notice how both make use of PI which is a constant Kotlin provides us, ready-made with π to as much precision as is possible by the computer

```
sqrt(2.0)
// 1.414213562373095
```

```
max(5, 10)
// 10
min(-5, -10)
// -10
```

```
max(sqrt(2.0), PI / 2)
// 1.570796326794897
```

Full list: https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.math/

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Naming data

- In your Kotlin code, you can give each piece of data a name you can use to refer to it later.
- The name carries with it an associated **type** that denotes what sort of data the name refers to, such as text, numbers, or a date

There are two kinds of data in Kotlin:

- Constants
- Variables

Constants

```
val number: Int = 10
```

This uses the **`val`** keyword to declare a constant called number which is of type **`Int`**.
Then it sets the value of the constant to the number 10.

The type **`Int`** can store integers. The way you store decimal numbers is like so:

```
val pi: Double = 3.14159
```

This time, the constant is a `Double`, a type that can store decimals with high precision.

*There's also a type called **`Float`**, short for floating point, that stores decimals with lower precision than **`Double`***

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Variables

- When you know you'll need to change some data, you should use a variable to represent that data instead of a constant.
- You declare a variable in a similar way, like so:

```
var variableNumber: Int = 42
```

- Once you've declared a variable, you're free to change it to whatever you wish, as long as the type remains the same.
- For example, to change the variable declared above, you could do this:

```
variableNumber = 0
variableNumber = 1_000_000
```

Using variables

```
fun main() {
    print("Input a number:")
    val x = readLine()!!.toInt()
    println("Your number is $x")
}
```

Using meaningful names

Good names can act as documentation and make your code easy to read.

A good name *specifically* describes the role of variable or constant. Here are some examples of good names:

- personAge
- numberOfPeople
- gradePointAverage

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Using meaningful names

Often a bad name is simply not descriptive enough. Here are some examples of bad names:

- a
- temp
- average

The key is to ensure that you'll understand what the variable or constant refers to when you read it again later.
In Kotlin, it is common to **camel case** names.

Camel Case

For variables and constants, follow these rules to properly case your names:

- Start with a lowercase letter.
- If the name is made up of multiple words, join them together and start every other word with an uppercase letter.
- If one of these words is an abbreviation, write the entire abbreviation in the same case (e.g., sourceURL and urlDescription)

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Increment and decrement

- A common operation that you will need is to be able to increment or decrement a variable.
- In Kotlin, this is achieved like so:

```
var counter: Int = 0
counter += 1
// counter = 1
counter -= 1
// counter = 0
```

```
var counter: Int = 0
counter = counter + 1
counter = counter - 1
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Demo

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Questions?

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Algorithms & Programming

(p.1 - basics)

Yevhen Berkunskyi, NUoS
eugeny.berkunsky@gmail.com
http://www.berkut.mk.ua