LECTURE 4: OPERATIONS

BY DR. AHMAD BARGHASH & ENG. LINA HAMMAD

Kotlin has a set of operators to perform arithmetic, assignment, comparison operators and more. You will learn to use these operators in this class.

ESCAPE CHARACTERS SUPPORTED IN KOTLIN

- \t - Inserts tab
- \b - Inserts backspace
- \n - Inserts newline
- \r - Inserts carriage return
- \' - Inserts single quote character
- \" - Inserts double quote character
- \\ - Inserts backslash
- \$ - Inserts dollar character

ESCAPED STRING AND RAW STRING

- Strings declared in double quotes can have escaped characters like '\n' (new line), '\t' (tab).

```
fun main() {
    var myEscapedString = "Hello Reader,\nWelcome to my Blog"
    println(myEscapedString)

}
```

The output is:

```
Hello Reader,
Welcome to my Blog
```

ESCAPED STRING AND RAW STRING

- In Kotlin, you also have an option to declare raw strings. These Strings have no escaping and can span multiple lines.

```
    var myMultilineRawString = """
    The Quick Brown Fox
    Jumped Over a Lazy Dog.
"""
    println(myMultilineRawString)
```

The output is:

```
 The Quick Brown Fox
Jumped Over a Lazy Dog.
```

USEFUL SHORTCUTS

- To format your code press:

  Alt + Ctrl + L (Windows)

  Option + Cmd + L (Mac)

- To show the type of variable click on it and press:

  Ctrl + Shift + P

- We can get to the source code. For example, when we type Int anywhere in the workspace then click on it and press:

  Ctrl + B (Windows)

  Cmd + B (Mac)

OPERATIONS ON NUMERIC TYPES

- Like other languages, Kotlin provides various operators to perform computations on numbers:
  1. Arithmetic operators (+, -, *, /, %)
  2. Comparison operators (==, !=, <, >, <=, >=)
  3. Assignment operators (+=, -=, *=, /=, %=)
  4. Increment & Decrement operators (++, --)

ARITHMETIC OPERATORS

- Here's a list of arithmetic operators in Kotlin:

Kotlin Arithmetic Operators

Operator	Meaning
+	Addition (also used for string concatenation)
-	Subtraction Operator
*	Multiplication Operator
/	Division Operator
%	Modulus Operator

ARITHMETIC OPERATORS - EXAMPLE

```
val number1 = 12.5
val number2 = 3.5

var result: Double

result = number1 + number2 // Addition
println("number1 + number2 = $result")

result = number1 - number2 // Subtraction
println("number1 - number2 = $result")

result = number1 * number2 // Multiplication
println("number1 * number2 = $result")

result = number1 / number2 // Division
println("number1 / number2 = $result")

result = number1 % number2 // Modulus
println("number1 % number2 = $result")
```

The output is:

```
number1 + number2 = 16.0
number1 - number2 = 9.0
number1 * number2 = 43.75
number1 / number2 = 3.5714285714285716
number1 % number2 = 2.0
```

HOW OPERATORS WORK IN KOTLIN

- Everything in Kotlin is an **object**, even the basic data types like Int, Char, Double, Boolean etc. Kotlin doesn't have separate primitive types and their corresponding boxed types like Java. (Boxing is wrapping a primitive type in an object so that it gets the object behaviour)
- Note that Kotlin may represent basic types like Int, Char, Boolean etc. as primitive values at runtime to improve performance, but for the end users, all of them are objects.
- Since all the data types are objects, the operations on these types are internally represented as function calls.
- For example, the addition operation a + b between two numbers a and b is represented as a function call a.plus(b).

HOW OPERATORS WORK IN KOTLIN

- Here's a table of arithmetic operators and their corresponding functions:

Expression	Function name	Translates to
a + b	plus	a.plus(b)
a - b	minus	a.minus(b)
a * b	times	a.times(b)
a / b	div	a.div(b)
a % b	mod	a.mod(b)

BASIC OPERATIONS -- EXAMPLE

- Solve the following using the operator methods in one line of code.

- If you start with 2 fish, and they breed twice, producing 71 offspring the first time, and 233 offspring the second time, and then 13 fish are swallowed by a hungry moray eel, how many fish do you have left? How many aquariums do you need if you can put 30 fish per aquarium?
  - Hint: You can chain method calls.
  - Hint: You can call the methods on numbers, and Kotlin will convert them to objects for you.
  - Bonus question: What is special about all the numbers of fish?

How many fish do you have left? `2.plus(71).plus(233).minus(13)`

How many aquariums do you need if you can put 30 fish per aquarium? `2.plus(71).plus(233).minus(13).div(30).plus(1)`

COMPARISON AND EQUALITY OPERATORS

- Here's a table of equality and comparison operators, their meaning, and corresponding functions:

Operator	Meaning	Expression	Translates to
>	greater than	a > b	a.compareTo(b) > 0
<	less than	a < b	a.compareTo(b) < 0
>=	greater than or equals to	a >= b	a.compareTo(b) >= 0
<=	less than or equals to	a < = b	a.compareTo(b) <= 0
==	is equal to	a == b	a?.equals(b) ?: (b === null)
!=	not equal to	a != b	!(a?.equals(b) ?: (b === null))

- Referential equality '===' operator is used to compare the reference of two variable or object. It will only be true if both the objects or variables pointing to the same object. The negated counterpart of === in Kotlin is !== which is used to compare if both the values are not equal to each other.tors, their meaning, and corresponding functions.

COMPARISON AND EQUALITY OPERATORS-- EXAMPLE

```
fun main() {
    var a = 10
    var b = 20
    var c = ((a + b) * (a + b)) / 2   // 450

    println("a = $a")
    println("b = $b")
    println("c = $c")

    var isALessThanB = a < b   // true
    println("isALessThanB = $isALessThanB")

    a++     // a now becomes 11
    b += 5  // b equals to 25 now
    println("a = $a")
    println("b = $b")

}
```

The output is:

```
a = 10
b = 20
c = 450
isALessThanB = true
a = 11
b = 25
```

ASSIGNMENT OPERATORS

- Assignment operators are used to assign value to a variable. We have already used simple assignment operator = before.

```
val age = 5
```

- Here, 5 is assigned to variable age using = operator.

- A shorthand operator is a shorter way to express something that is already available in the Kotlin programming language.
- Shorthand operators are +=, -=, *=, /= and *=

ASSIGNMENT OPERATORS

- Here's a list of all assignment operators and their corresponding functions:

Expression	Equivalent to	Translates to
a +=b	a = a + b	a.plusAssign(b)
a -= b	a = a - b	a.minusAssign(b)
a *= b	a = a * b	a.timesAssign(b)
a /= b	a = a / b	a.divAssign(b)
a %= b	a = a % b	a.modAssign(b)

ASSIGNMENT OPERATORS -- EXAMPLE

```
fun main() {
    var number = 12
    number *= 5   // number = number*5
    println("number  = $number")
}
```

The output is:

```
number = 60
```

UNARY PREFIX AND INCREMENT / DECREMENT OPERATORS

- Here's a table of unary operators, their meaning, and corresponding functions:

Operator	Meaning	Expression	Translates to
+	Unary plus	+a	a.unaryPlus()
-	Unary minus (inverts sign)	-a	a.unaryMinus()
!	not (inverts value)	!a	a.not()
++	Increment: increases value by1	++a	a.inc()
--	Decrement: decreases value by 1	--a	a.dec()

INCREMENT AND DECREMENT OPERATOR

- Increment Operator:
  - It is used to increment a value by 1. There are two varieties of increment operator:
    - **Post-Increment :** Value is first used for computing the result and then incremented. Ex: a++
    - **Pre-Increment :** Value is incremented first and then result is computed. Ex: ++a

- Decrement operator:
  - It is used for decrementing the value by 1. There are two varieties of decrement operator.
    - **Post-decrement :** Value is first used for computing the result and then decremented. Ex: a--
    - **Pre-decrement :** Value is decremented first and then result is computed. Ex: --a

INCREMENT AND DECREMENT OPERATOR

```
var variable_one = 2
var variable_two = ++variable_one //Pre-Increment Operator

println("variable_one = $variable_one")
println("variable_two = $variable_two")
```

The output is:

```
variable_one = 3
variable_two = 3
```

```
var variable_one = 2
var variable_two = variable_one++ //Post-Increment Operator

println("variable_one = $variable_one")
println("variable_two = $variable_two")
```

The output is:

```
variable_one = 3
variable_two = 2
```

INCREMENT AND DECREMENT OPERATOR

```
fun main(){
    var num1 = 10
    var num2 = ++num1 //Pre-Increment Operator
    var num3 = num2++ //Post-Increment Operator

    println("num1 = $num1")
    println("num2 = $num2")
    println("num3 = $num3")
}
```

The output is:

```
num1 = 11

num2 = 12

num3 = 11
```

INCREMENT / DECREMENT OPERATORS EXAMPLE

```
fun main(args: Array<String>) {
    val a = 1
    val b = true
    var c = 1

    var result: Int
    var booleanResult: Boolean

    result = -a
    println("-a = $result")

    booleanResult = !b
    println("!b = $booleanResult")

    --c
    println("--c = $c")
}
```

The output is:

```
-a = -1
!b = false
--c = 0
```

ALL TOGETHER

- All the operators that we looked at in the previous section have a symbolic name which is used to translate any expression containing those operators into the corresponding function call.

Expression	Translates to
a + b	a.plus(b)
a - b	a.minus(b)
a * b	a.times(b)
a / b	a.div(b)
a % b	a.rem(b)
a++	a.inc()
a--	a.dec()
a > b	a.compareTo(b) > 0
a < b	a.compareTo(b) < 0
a += b	a.plusAssign(b)
...	...

- You can check out other expressions and their corresponding function calls on Kotlin's reference page.

OPERATIONS ON BOOLEAN TYPES

- Kotlin supports following logical operators for performing operations on boolean types:
  1. || - Logical OR
  2. && - Logical AND
  3. ! - Logical NOT
- Here's a table of logical operators, their meaning, and corresponding functions.

Operator	Description	Expression	Corresponding Function
||	`true` if either of the Boolean expression is `true`	`(a>b)||(a<c)`	`(a>b)or(a<c)`
&&	true if all Boolean expressions are `true`	`(a>b)&&(a<c)`	`(a>b)and(a<c)`

OPERATIONS ON BOOLEAN TYPES

- Or truth table

a	b	a or b
T	T	T
F	T	T
T	F	T
F	F	F

- And truth table

a	b	a and b
T	T	T
F	T	F
T	F	F
F	F	F

- Not truth table

a	!a
T	F
F	T

OPERATIONS ON BOOLEAN TYPES -- EXAMPLE

```
fun main() {
    var a = true
    var b = false

    println(a && b)
    println(a || b)

}
```

The output is:

```
false
true
```

```
fun main() {
    val a = 10
    val b = 9
    val c = -1
    val result: Boolean

    // result is true is a is largest
    result = (a>b) && (a>c) // result = (a>b) and (a>c)
    println(result)
}
```

The output is:

```
true
```

OPERATIONS ON STRINGS

- Do not miss: the + operator is also used for the concatenation of String values.

```
fun main() {
    var firstName = "Rajeev"
    var lastName = "Singh"
    var fullName = firstName + " " + lastName
    println(fullName)
}
```

The output is:

```
Rajeev Singh
```

EXAMPLE: CONCATENATION OF STRINGS

```
fun main() {
    val start = "Talk is cheap. "
    val middle = "Show me the code. "
    val end = "- Linus Torvalds"
    val result = start + middle + end
    println(result)
}
```

The output is:

```
Talk is cheap. Show me the code. - Linus Torvalds
```

PRECEDENCE ORDER

- When two operators share an operand the operator with the higher *precedence* goes first. For example, 1 + 2 * 3 is treated as 1 + (2 * 3), whereas 1 * 2 + 3 is treated as (1 * 2) + 3 since multiplication has a higher precedence than addition.

Level	Operator	Description	Associativity
16	()	parentheses	left to right
15	++ --	unary increment unary decrement	not associative
14	+ -	unary plus unary minus	right to left
12	* / %	Multiplicative	left to right
11	+ - +	additive string concatenation	left to right
9	< <= > >=	Relational	not associative
8	== !=	equality	left to right
4	&&	logical AND	left to right
3	||	logical OR	left to right
1	= += -= *= /= %=	assignment	right to left

LOGICAL OPERATION -- EXAMPLE

```
fun main() {
    var a = 5;
    var b = 8;
    var year = 2019
    var c = (2 == 2) && (4 != 5)
    var d = 4 > 5 && 2 < 7
    var e = !(7 > 12 || 14 < 18)
    var f = a.plus(b)
    var g = ((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0)
    var h = (a+7) > (b++ + 7) && ((a % 2) != (b/2)) != true


    println ("a = $a")
    println ("b = $b")
    println ("c = $c")
    println ("d = $d")
    println ("e = $e")
    println ("f = $f")
    println ("g = $g")
    println ("h = $h")
}
```

The output is:

```
a = 5
b = 9
c = true
d = false
e = false
f = 13
g = false
h = false
```