Operators in Kotlin

Operators are used to perform operations on operands(value and variable). There are following operators in Kotlin-

- Arithmetic Operators
- Assignment Operators
- Logical operators
- Comparison and Equality Operators
- Unary prefix and Increment/Decrement
- 'In' operator
- Indexed access operator
- Invoke operator

1.

Arithmetic Operators

If you want to perform basic algebraic operations like +, -, * and / in Kotlin then use arithmetic operators.

Operator	Usage	Description	Translates to
+	value1 + value2	Adds value1 and value2.	value1.plus(value2)
-	value1 - value2	Subtract value2 from value1.	value1.minus(value2)
*	value1 * value2	Multiplies value1 by value2.	value1.times(value2)
/	value1 / value2	Divides value1 by value2.	value1.div(value2)
%	value1 % value2	Calculates the remainder of dividing value1 by value2.	value1.mod(value2)

Example of Arithmetic Operators

```
fun main(args: Array) {

    val num1 = 200
    val num2 = 10
    var result: Double

    result = num1 + num2
    println("Addition = $result")

    result = num1 - num2
    println("Subtraction = $result")

    result = num1 * num2
    println("Multiplication = $result")

    result = num1 / num2
    println("Division = $result")

    result = num1 % num2
    println("Remainder = $result")
}
```

Output

```
Addition = 210
Subtraction = 190
Multiplication = 2000
Division = 20
Remainder = 0
```

2.

Assignment Operators

The assignment operator is used to assign the value to a variable or operand. Let's see how we can use it-

```
val i = 10
```

Here, 10 is assigned to variable i using `=` operator.

Operator	Usage	Description	Translates to
+=	value1 += value2	value1 = value1 + value2	value1.plusAssign(value2)
-=	value1 -= value2	value1 = value1 - value2	value1.minusAssign(value2)

Operator	Usage	Description	Translates to
*=	value1 *= value2	value1 = value1 * value2	value1.timesAssign(value2)
/=	value1 /= value2	value1 = value1 / value2	value1.divAssign(value2)
%=	value1 %= value2	value1 = value1 % value2	value1.modAssign(value2)

Example in Assignment Operators

```
fun main(args: Array) {
    var num = 15

    number += 5    // num = num+5
    println("num  = $num")
}
```

Output

```
20
```

3.

Logical Operators

Logical operators are used in control flow such as if expression, when expression, and loop. There two types of logical operators in Kotlin- `||(Logical OR)` and `&&(Logical AND)`.

Operator	Usage	Description
Logical OR(||)	expression1 || expression2	It checks the second operand when the first operand is false. If the first operand evaluates to true, then the second operand is not even checked.
Logical AND(&&)	expression1 && expression2	It checks the second operand when the first operand turns out to be true. If the first operand evaluates to false, then the second operand is not even checked.

Example of Logical Operator

```
fun main(args: Array){
    val p = 5
    val q = 8
    val r = -2
    val result: Boolean

    // result is true is a is largest
    result = (p>q) && (p>r) // result = (p>q) and (p>r)
    println(result)
}
```

Output

```
true
```

6. 'In' Operator

The operator `in` is used to check whether an object belongs to a collection.

Operator	Expression	Translates to
in	value1 in value2	value2.contains(value1)
!in	value1 !in value2	!value2.contains(value1)

Example of In Operator

```
fun main(args: Array){
    val numbers = intArrayOf(2, 4, 40, -3)

    if (4 in numbers){
        println("numbers array contains 4.")
    }
}
```

Output

```
numbers array contains 4.
```

7. Index access Operator

Expression	Translates to
`a[i]`	`a.get(i)`

Expression	Translates to
`a[i, n]`	`a.get(i, n)`
`a[i1, i2, ..., in]`	`a.get(i1, i2, ..., in)`
`a[i] = b`	`a.set(i, b)`
`a[i, n] = b`	`a.set(i, n, b)`
`a[i1, i2, ..., in] = b`	`a.set(i1, i2, ..., in, b)`

Example of In Operator

```
fun main(args: Array){

    val p  = intArrayOf(1, 2, 3, 4, - 1)
    println(p[1])
    p[1]= 12
    println(p[1])
}
```

Output

```
2
12
```

8. Invoke Operator

Expression	Translates to
`a()`	`a.invoke()`
`a(i)`	`a.invoke(i)`
`a(i1, i2, ..., in)`	`a.inkove(i1, i2, ..., in)`
`a[i] = b`	`a.set(i, b)`

Parentheses are translated to calls to `invoke` number of arguments.