Washtenaw Community College
ANN ARBOR, MICHIGAN

CPS251

Android Development

by Scott Shaper

Kotlin Operators

Think of operators in programming like the symbols you use in math class. Just as you use +, -, *, and ÷ to perform calculations, programming operators help you manipulate data, compare values, and control the flow of your program. Understanding operators is crucial for writing expressions and making decisions in your code.

In this lesson, we'll explore the different types of operators in Kotlin: arithmetic operators for calculations, comparison operators for making decisions, logical operators for combining conditions, and assignment operators for updating variables. These are the building blocks that make your code dynamic and responsive.

Quick Reference Table

Operator Type	Description	Common Use
Arithmetic	Perform mathematical calculations	Calculations, math operations
Comparison	Compare values and return true/false	Making decisions, conditions
Logical	Combine multiple conditions	Complex decision making
Assignment	Update variable values	Storing results, updating data

Arithmetic Operators

When to Use

- When you need to perform mathematical calculations
- When working with numbers in your program
- When calculating totals, averages, or other numeric results

Operator	What It Does	Example
+	Addition	5 + 3 = 8
-	Subtraction	10 - 4 = 6
*	Multiplication	6 * 7 = 42
/	Division	15 / 3 = 5
%	Modulo (remainder)	17 % 5 = 2

Practical Examples

Copy Code

```
// ==========================================
// BASIC ARITHMETIC OPERATIONS
// ==========================================

// Example 1: Simple calculations
// These demonstrate the basic arithmetic operators
val a = 15
val b = 4

println("=== BASIC ARITHMETIC ===")
println("a = $a, b = $b")
println("Addition: $a + $b = ${a + b}")
println("Subtraction: $a - $b = ${a - b}")
```

```
println("Multiplication: $a * $b = ${a * b}")
println("Division: $a / $b = ${a / b}")
println("Modulo: $a % $b = ${a % b}")

// Example 2: Real-world calculations
// This shows practical applications of arithmetic
println("\n=== REAL-WORLD CALCULATIONS ===")

// Shopping cart calculations
val itemPrice = 29.99
val quantity = 3
val taxRate = 0.08  // 8% tax
val discount = 0.10  // 10% discount

val subtotal = itemPrice * quantity
val discountAmount = subtotal * discount
val discountedSubtotal = subtotal - discountAmount
val taxAmount = discountedSubtotal * taxRate
val finalTotal = discountedSubtotal + taxAmount

println("Shopping Cart Calculation:")
println("  Item price: $${String.format("%.2f", itemPrice)}")
println("  Quantity: $quantity")
println("  Subtotal: $${String.format("%.2f", subtotal)}")
println("  Discount (10%): $${String.format("%.2f", discountAmount)}")
println("  After discount: $${String.format("%.2f", discountedSubtotal)}")
println("  Tax (8%): $${String.format("%.2f", taxAmount)}")
println("  Final total: $${String.format("%.2f", finalTotal)}")

// Example 3: Mathematical formulas
// This demonstrates using arithmetic in formulas
println("\n=== MATHEMATICAL FORMULAS ===")

// Circle calculations
```

```
val radius = 5.0
val pi = 3.14159

val circumference = 2 * pi * radius
val area = pi * radius * radius

println("Circle with radius $radius:")
println("  Circumference: ${String.format("%.2f", circumference)}")
println("  Area: ${String.format("%.2f", area)}")

// Temperature conversion
val celsius = 25.0
val fahrenheit = (celsius * 9/5) + 32

println("Temperature conversion:")
println("  Celsius: $celsius°C")
println("  Fahrenheit: ${String.format("%.1f", fahrenheit)}°F")

// Example 4: Working with different data types
// This shows how arithmetic works with different types
println("\n=== DATA TYPE ARITHMETIC ===")

val intValue = 10
val doubleValue = 3.5
val longValue = 100L

// Int arithmetic
println("Int arithmetic:")
println("  $intValue + 5 = ${intValue + 5}")
println("  $intValue * 2 = ${intValue * 2}")

// Double arithmetic
println("Double arithmetic:")
println("  $doubleValue + 2.5 = ${doubleValue + 2.5}")
```

```
println("  $doubleValue * 3 = ${doubleValue * 3}")

// Mixed type arithmetic
println("Mixed type arithmetic:")
println("  $intValue + $doubleValue = ${intValue + doubleValue}")
println("  $intValue * $longValue = ${intValue * longValue}")

// Example 5: Modulo operations
// This demonstrates the modulo operator for remainders
println("\n=== MODULO OPERATIONS ===")

val numbers = listOf(7, 15, 23, 30, 42)
val divisor = 5

for (number in numbers) {
    val remainder = number % divisor
    val quotient = number / divisor
    println("$number ÷ $divisor = $quotient remainder $remainder")
}

// Check if numbers are even or odd
println("\nEven/Odd check:")
for (number in numbers) {
    val isEven = number % 2 == 0
    println("$number is ${if (isEven) "even" else "odd"}")
}
```

Comparison Operators

When to Use

- When you need to compare values
- When making decisions in your code
- When checking if conditions are true or false

Operator	What It Does	Example
`==`	Equal to	`5 == 5` returns `true`
`!=`	Not equal to	`5 != 3` returns `true`
`>`	Greater than	`10 > 5` returns `true`
`<`	Less than	`3 < 7` returns `true`
`>=`	Greater than or equal to	`5 >= 5` returns `true`
`<=`	Less than or equal to	`4 <= 6` returns `true`

Practical Examples

Copy Code

```
// ==========================================
// COMPARISON OPERATOR EXAMPLES
// ==========================================

// Example 1: Basic comparisons
// These demonstrate the fundamental comparison operators
println("=== BASIC COMPARISONS ===")

val x = 10
val y = 5
val z = 10

println("x = $x, y = $y, z = $z")
println("x == y: ${x == y}")      // Equal to
println("x != y: ${x != y}")      // Not equal to
println("x > y: ${x > y}")        // Greater than
println("x < y: ${x < y}")        // Less than
println("x >= z: ${x >= z}")      // Greater than or equal to
println("x <= z: ${x <= z}")      // Less than or equal to
```

```
// Example 2: String comparisons
// This shows how comparison works with text
println("\n=== STRING COMPARISONS ===")

val name1 = "Alice"
val name2 = "Bob"
val name3 = "Alice"

println("name1 = '$name1', name2 = '$name2', name3 = '$name3'")
println("name1 == name2: ${name1 == name2}")
println("name1 == name3: ${name1 == name3}")
println("name1 != name2: ${name1 != name2}")

// String ordering (alphabetical)
println("name1 < name2: ${name1 < name2}")  // 'Alice' comes before 'Bob'
println("name2 > name1: ${name2 > name1}")  // 'Bob' comes after 'Alice'

// Example 3: Real-world comparison scenarios
// This demonstrates practical uses of comparisons
println("\n=== REAL-WORLD COMPARISONS ===")

// Age verification system
val userAge = 17
val minimumAge = 18
val seniorAge = 65

val canVote = userAge >= minimumAge
val isMinor = userAge < minimumAge
val isSenior = userAge >= seniorAge

println("Age verification for user age $userAge:")
println("  Can vote: $canVote")
println("  Is minor: $isMinor")
```

```
println("  Is senior: $isSenior")

// Grade calculation
val score = 85
val passingScore = 60
val excellentScore = 90

val isPassing = score >= passingScore
val isExcellent = score >= excellentScore
val needsImprovement = score < passingScore

println("\nGrade analysis for score $score:")
println("  Is passing: $isPassing")
println("  Is excellent: $isExcellent")
println("  Needs improvement: $needsImprovement")

// Example 4: Range checking
// This shows how to check if values fall within ranges
println("\n=== RANGE CHECKING ===")

val temperature = 72
val lowTemp = 65
val highTemp = 78

val isComfortable = temperature >= lowTemp && temperature <= highTemp
val isTooCold = temperature < lowTemp
val isTooHot = temperature > highTemp

println("Temperature analysis for $temperature°F:")
println("  Is comfortable: $isComfortable")
println("  Is too cold: $isTooCold")
println("  Is too hot: $isTooHot")

// Example 5: Multiple comparisons
```

```
// This demonstrates combining multiple comparison operations
println("\n=== MULTIPLE COMPARISONS ===")

val testScores = listOf(95, 87, 72, 100, 58, 89)

for (score in testScores) {
    val grade = when {
        score >= 90 -> "A"
        score >= 80 -> "B"
        score >= 70 -> "C"
        score >= 60 -> "D"
        else -> "F"
    }

    val status = when {
        score == 100 -> "Perfect!"
        score >= 90 -> "Excellent"
        score >= 80 -> "Good"
        score >= 70 -> "Satisfactory"
        score >= 60 -> "Needs improvement"
        else -> "Failed"
    }

    println("Score $score: Grade $grade - $status")
}
```

Logical Operators

When to Use

- When you need to combine multiple conditions
- When creating complex decision logic
- When you need to check if multiple things are true or false

Operator	What It Does	Example
`&&`	Logical AND (both must be true)	`true && true` returns `true`
`||`	Logical OR (either can be true)	`true || false` returns `true`
`!`	Logical NOT (inverts the value)	`!true` returns `false`

Practical Examples

Copy Code

```
// ========================================
// LOGICAL OPERATOR EXAMPLES
// ========================================

// Example 1: Basic logical operations
// These demonstrate the fundamental logical operators
println("=== BASIC LOGICAL OPERATIONS ===")

val isSunny = true
val isWarm = true
val isRaining = false

println("Weather conditions:")
println("  Is sunny: $isSunny")
println("  Is warm: $isWarm")
println("  Is raining: $isRaining")

// Logical AND - both conditions must be true
val isGoodWeather = isSunny && isWarm
println("  Is good weather (sunny AND warm): $isGoodWeather")

// Logical OR - either condition can be true
val isOutdoorWeather = isSunny || isWarm
```

```
println("  Is outdoor weather (sunny OR warm): $isOutdoorWeather")

// Logical NOT - inverts the value
val isNotRaining = !isRaining
println("  Is not raining: $isNotRaining")

// Example 2: Complex logical expressions
// This shows how to combine multiple logical operations
println("\n=== COMPLEX LOGICAL EXPRESSIONS ===")

// User authentication system
val hasValidUsername = true
val hasValidPassword = true
val isAccountActive = false
val isNotLocked = true

// User can login if they have valid credentials AND account is active AND not locked
val canLogin = hasValidUsername && hasValidPassword && isAccountActive &&
isNotLocked

// User can reset password if they have valid username OR if account is locked
val canResetPassword = hasValidUsername || !isNotLocked

println("Authentication status:")
println("  Has valid username: $hasValidUsername")
println("  Has valid password: $hasValidPassword")
println("  Is account active: $isAccountActive")
println("  Is not locked: $isNotLocked")
println("  Can login: $canLogin")
println("  Can reset password: $canResetPassword")

// Example 3: Real-world logical scenarios
// This demonstrates practical applications of logical operators
println("\n=== REAL-WORLD LOGICAL SCENARIOS ===")
```

```
// Shopping cart validation
val hasItems = true
val hasValidPayment = true
val isWithinBudget = false
val hasShippingAddress = true

// Order can be placed if all conditions are met
val canPlaceOrder = hasItems && hasValidPayment && isWithinBudget &&
hasShippingAddress

// Order can be saved for later if it has items but doesn't meet other criteria
val canSaveForLater = hasItems && (!hasValidPayment || !isWithinBudget ||
!hasShippingAddress)

println("Shopping cart status:")
println("  Has items: $hasItems")
println("  Has valid payment: $hasValidPayment")
println("  Is within budget: $isWithinBudget")
println("  Has shipping address: $hasShippingAddress")
println("  Can place order: $canPlaceOrder")
println("  Can save for later: $canSaveForLater")

// Example 4: De Morgan's Law demonstration
// This shows how logical expressions can be rewritten
println("\n=== DE MORGAN'S LAW DEMONSTRATION ===")

val a = true
val b = false

// Original expression: !(a && b)
val original = !(a && b)

// Using De Morgan's Law: !a || !b
```

```
val demorgan = !a || !b

println("De Morgan's Law demonstration:")
println("  a = $a, b = $b")
println("  Original: !(a && b) = $original")
println("  De Morgan: !a || !b = $demorgan")
println("  Are they equal? ${original == demorgan}")

// Example 5: Short-circuit evaluation
// This demonstrates how logical operators can short-circuit
println("\n=== SHORT-CIRCUIT EVALUATION ===")

fun checkFirst(): Boolean {
    println("  Checking first condition...")
    return false
}

fun checkSecond(): Boolean {
    println("  Checking second condition...")
    return true
}

fun checkThird(): Boolean {
    println("  Checking third condition...")
    return true
}

println("Short-circuit with AND (&&):")
val result1 = checkFirst() && checkSecond() && checkThird()
println("  Final result: $result1")

println("\nShort-circuit with OR (||):")
```

```
val result2 = checkFirst() || checkSecond() || checkThird()
println("  Final result: $result2")
```

Assignment Operators

When to Use

- When you need to update variable values
- When performing calculations and storing results
- When incrementing or decrementing values

Operator	What It Does	Example
`=`	Simple assignment	`x = 5`
`+=`	Add and assign	`x += 3` is same as `x = x + 3`
`-=`	Subtract and assign	`x -= 2` is same as `x = x - 2`
`*=`	Multiply and assign	`x *= 4` is same as `x = x * 4`
`/=`	Divide and assign	`x /= 2` is same as `x = x / 2`
`%=`	Modulo and assign	`x %= 3` is same as `x = x % 3`

Practical Examples

Copy Code

```
// ========================================
// ASSIGNMENT OPERATOR EXAMPLES
// ========================================

// Example 1: Basic assignment operators
// These demonstrate the fundamental assignment operations
println("=== BASIC ASSIGNMENT OPERATORS ===")
```

```
var number = 10
println("Initial value: $number")

// Simple assignment
number = 15
println("After simple assignment: $number")

// Add and assign
number += 5
println("After += 5: $number")

// Subtract and assign
number -= 3
println("After -= 3: $number")

// Multiply and assign
number *= 2
println("After *= 2: $number")

// Divide and assign
number /= 4
println("After /= 4: $number")

// Modulo and assign
number %= 3
println("After %= 3: $number")

// Example 2: Counter and accumulator patterns
// This shows common programming patterns using assignment operators
println("\n=== COUNTER AND ACCUMULATOR PATTERNS ===")

// Counter pattern
var counter = 0
println("Counter pattern:")
```

```
for (i in 1..5) {
    counter += 1
    println("  Iteration $i: counter = $counter")
}

// Accumulator pattern
var total = 0
val numbers = listOf(10, 20, 30, 40, 50)
println("\nAccumulator pattern:")
for (number in numbers) {
    total += number
    println("  Added $number, total = $total")
}

// Running average
var sum = 0.0
var count = 0
val scores = listOf(85, 92, 78, 96, 88)

println("\nRunning average calculation:")
for (score in scores) {
    sum += score
    count += 1
    val average = sum / count
    println("  Score: $score, Running average: ${String.format("%.1f", average)}")
}

// Example 3: Real-world assignment scenarios
// This demonstrates practical uses of assignment operators
println("\n=== REAL-WORLD ASSIGNMENT SCENARIOS ===")

// Bank account balance management
var balance = 1000.0
val transactions = listOf(150.0, -75.0, 200.0, -50.0, 300.0)
```

```
println("Bank account transactions:")
println("  Initial balance: $${String.format("%.2f", balance)}")

for (transaction in transactions) {
    if (transaction > 0) {
        balance += transaction
        println("  Deposited $${String.format("%.2f", transaction)}, new balance:
$${String.format("%.2f", balance)}")
    } else {
        balance += transaction  // transaction is negative, so this subtracts
        println("  Withdrew $${String.format("%.2f", -transaction)}, new balance:
$${String.format("%.2f", balance)}")
    }
}

// Example 4: Increment and decrement operations
// This shows how to increase or decrease values
println("\n=== INCREMENT AND DECREMENT ===")

var count = 0
println("Increment operations:")
count += 1
println("  count += 1: $count")
count += 1
println("  count += 1: $count")

var temperature = 72
println("\nTemperature adjustments:")
temperature += 5
println("  Temperature increased by 5: $temperature")
temperature -= 3
println("  Temperature decreased by 3: $temperature")
```

```
// Example 5: Compound assignment with different types
// This demonstrates assignment operators with various data types
println("\n=== COMPOUND ASSIGNMENT WITH DIFFERENT TYPES ===")

// String concatenation
var message = "Hello"
message += " World"
message += "!"
println("String concatenation: $message")

// List operations
var numbers = mutableListOf(1, 2, 3)
numbers += 4
numbers += 5
println("List addition: $numbers")

// Boolean operations
var isActive = true
isActive = isActive && false
println("Boolean assignment: $isActive")
```

Tips for Success

- Remember that comparison operators return boolean values (true/false)
- Use parentheses to clarify the order of operations in complex expressions
- Be careful with floating-point comparisons - use approximate equality when needed
- Understand short-circuit evaluation for logical operators
- Use assignment operators to make your code more concise

Common Mistakes to Avoid

- Using `=` instead of `==` for comparisons
- Forgetting that string comparison uses `==` in Kotlin (not `.equals()`)

- Not considering operator precedence in complex expressions
- Using logical operators when you need bitwise operators
- Forgetting that assignment operators modify the original variable

Best Practices

- Use parentheses to make complex expressions more readable
- Choose meaningful variable names to make expressions clear
- Break complex logical expressions into smaller, more readable parts
- Use assignment operators to make your code more concise
- Test your logical expressions with different input values