# Lý thuyết Kotlin

## 1. Tổng quan về Kotlin

Kotlin là một ngôn ngữ lập trình hiện đại, hướng đối tượng và hỗ trợ lập trình hàm được phát triển bởi JetBrains, công ty nổi tiếng với các IDE như IntelliJ. Kotlin được thiết kế để tương thích hoàn toàn với Java, cho phép các nhà phát triển dễ dàng tích hợp Kotlin vào các dự án Java hiện có. Ngôn ngữ này được Google chính thức hỗ trợ cho phát triển ứng dụng Android kể từ năm 2017.

## 2. Đặc điểm chính của Kotlin

- **Hướng đối tượng và hỗ trợ lập trình hàm:** Kotlin kết hợp cả hai phong cách lập trình, cho phép bạn viết code theo cách hướng đối tượng hoặc hàm.
- **Tương thích với Java:** Kotlin có thể chạy trên JVM (Java Virtual Machine) và có thể tích hợp với mã Java.
- **An toàn giá trị null (giá trị đặc biệt):** Kotlin có hệ thống kiểu dữ liệu mạnh mẽ để xử lý giá trị null, giảm thiểu lỗi NullPointerException (một lỗi phổ biến trong Java khi bạn cố gắng truy cập một đối tượng null).
- **Đơn giản và rõ ràng:** Cú pháp của Kotlin được thiết kế để dễ đọc và viết, giúp tăng năng suất lập trình.
- **Hỗ trợ coroutines:** Kotlin cung cấp hỗ trợ mạnh mẽ cho lập trình đồng bộ và bất đồng bộ thông qua coroutines.
- **Cộng đồng và tài nguyên:** Kotlin có một cộng đồng lớn và nhiều tài nguyên học tập, bao gồm sách, khóa học, và tài liệu chính thức. Ví dụ: https://kotlinlang.org/

## 3. Cú pháp cơ bản của Kotlin

### 3.1 Khai báo biến

+ Biến không thay đổi - **immutable** (**val**): Một biến không thể thay đổi giá trị sau khi được khai báo.

+ Biến có thể thay đổi - **mutable** (**var**): Một biến có thể thay đổi giá trị sau khi được khai báo.

```
val name: String = "Kotlin"

var age: Int = 5
```

### 3.2 Kiểu dữ liệu

Kotlin cung cấp nhiều kiểu dữ liệu, bao gồm:

- **Số nguyên:** Byte, Short, Int, Long
- **Số thực:** Float, Double
- **Ký tự:** Char

- **Chuỗi:** String
- **Luận lý:** Boolean
- **Mảng:** Array

**Ví dụ:**

```
val byteValue: Byte = 100
val shortValue: Short = 1000
val intValue: Int = 100000
val longValue: Long = 10000000000L
val floatValue: Float = 3.14f
val doubleValue: Double = 3.141592653589793
val charValue: Char = 'A'
val stringValue: String = "Hello, Kotlin!"
val booleanValue: Boolean = true
val arrayValue: Array<Int> = arrayOf(1, 2, 3, 4, 5)
```

### 3.3 Hàm

Cú pháp khai báo hàm trong Kotlin:

```
fun greeting(name: String): String {
    return "Hello, $name!"
}
// Hàm không trả về giá trị
fun printGreeting(name: String) {
    println("Hello, $name!")
}
```

### 3.4 Lớp và đối tượng

Kotlin hỗ trợ lập trình hướng đối tượng với các lớp và đối tượng.

```
class Person(val name: String, var age: Int) {
    fun introduce() {
        println("My name is $name and I am $age years old.")
    }
}
```

```
fun main() {
    val person = Person("Alice", 30)
    person.introduce()
}
```

### 3.5 Interface và Inheritance

Kotlin hỗ trợ interface và kế thừa.

```
interface Drawable {
    fun draw()
}

class Circle : Drawable {
    override fun draw() {
        println("Drawing a circle")
    }
}
fun main() {
    val circle = Circle()
    circle.draw()
}
```

## 4. Tính năng nổi bật

### 4.1 An toàn về giá trị null

Kotlin cung cấp hệ thống kiểu dữ liệu mạnh mẽ để xử lý null, giúp giảm thiểu lỗi NullPointerException.

**Ví dụ:**

```
val nullableString: String? = null
val nonNullableString: String = nullableString ?: "Default Value"
```

### 4.2 Extension Functions

Kotlin cho phép bạn thêm các phương thức mới vào các lớp đã tồn tại mà không cần kế thừa hoặc sử dụng design patterns

**Ví dụ:**

```
fun String.lastChar(): Char {
    return this.get(this.length - 1)
}
fun main() {
    val str = "Kotlin"
    println(str.lastChar()) // Output: n
}
```

**4.3 Coroutines**

Kotlin cung cấp hỗ trợ mạnh mẽ cho lập trình đồng bộ và bất đồng bộ thông qua coroutines.

**Ví dụ:**

```
import kotlinx.coroutines.*
fun main() = runBlocking {
    launch {
        delay(1000L)
        println("World!")
    }
    println("Hello,")
}
```

**5. Ứng dụng của Kotlin**

- **Phát triển Android:** Kotlin là ngôn ngữ chính thức cho phát triển ứng dụng Android.
- **Phát triển web:** Kotlin có thể được sử dụng để phát triển backend và frontend thông qua Kotlin.
- **Phát triển đa nền tảng:** Kotlin Multiplatform cho phép bạn viết code một lần và sử dụng trên nhiều nền tảng như Android, iOS, web, và desktop.
- **Phát triển server-side:** Kotlin có thể được sử dụng với các framework như Spring Boot để phát triển ứng dụng server-side.

=> Kotlin là một ngôn ngữ lập trình hiện đại, mạnh mẽ và linh hoạt, phù hợp cho nhiều loại dự án. Với sự hỗ trợ mạnh mẽ từ JetBrains và Google, Kotlin đã trở thành một lựa chọn phổ biến cho các nhà phát triển.

**Kiểu dữ liệu và cách khai báo biến trong Kotlin**

**1. Giới thiệu về Kiểu dữ liệu trong Kotlin**

Kotlin là một ngôn ngữ lập trình hiện đại và hướng đối tượng, cung cấp nhiều kiểu dữ liệu để lưu trữ và xử lý thông tin. Mọi kiểu dữ liệu trong Kotlin đều là đối tượng, điều này giúp ngôn ngữ này trở nên mạnh mẽ và linh hoạt hơn. Dưới đây là một số kiểu dữ liệu phổ biến trong Kotlin:

- **Số nguyên (Integers):** Byte, Short, Int, Long
- **Số thực (Floating-point numbers):** Float, Double
- **Ký tự (Characters):** Char
- **Chuỗi (Strings):** String
- **Luận lý (Booleans):** Boolean
- **Mảng (Arrays):** Array

**2. Khai báo biến trong Kotlin**

Trong Kotlin, bạn có thể khai báo biến sử dụng từ khóa **val** (cho biến không đổi - **immutable**) và **var** (cho biến có thể thay đổi - **mutable**). Cú pháp khai báo biến như sau:

val <tên biến>: <kiểu dữ liệu> = <giá trị>

var <tên biến>: <kiểu dữ liệu> = <giá trị>

**3. Các kiểu dữ liệu cụ thể**

**3.1 Số nguyên (Integers)**

- **Byte:** 8 bits, giá trị từ -128 đến 127
- **Short:** 16 bits, giá trị từ -32,768 đến 32,767
- **Int:** 32 bits, giá trị từ -2,147,483,648 đến 2,147,483,647
- **Long:** 64 bits, giá trị từ -9,223,372,036,854,775,808 đến 9,223,372,036,854,775,807

**Ví dụ:**

```
val byteValue: Byte = 100

val shortValue: Short = 1000

val intValue: Int = 100000

val longValue: Long = 10000000000L
```

**3.2 Số thực (Floating-point numbers)**

- **Float:** 32 bits, độ chính xác thấp hơn Double

- **Double:** 64 bits, độ chính xác cao hơn Float

**Ví dụ:**

val floatValue: Float = 3.14f

val doubleValue: Double = 3.141592653589793

### 3.3 Ký tự (Characters)

Ký tự trong Kotlin được biểu diễn bằng từ khóa Char.

**Ví dụ:**

val charValue: Char = 'A'

### 3.4 Chuỗi (Strings)

Chuỗi trong Kotlin được biểu diễn bằng từ khóa String.

**Ví dụ:**

val stringValue: String = "Hello, Kotlin!"

### 3.5 Luận lý (Booleans)

Luận lý trong Kotlin được biểu diễn bằng từ khóa Boolean.

**Ví dụ:**

val booleanValue: Boolean = true

### 3.6 Mảng (Arrays)

Mảng trong Kotlin được biểu diễn bằng từ khóa Array.

**Ví dụ:**

val arrayValue: Array<Int> = arrayOf(1, 2, 3, 4, 5)

**=> Lưu ý: Tối ưu hóa hệ thống bằng cách chọn kiểu dữ liệu phù hợp**

Việc chọn kiểu dữ liệu phù hợp không chỉ giúp code của bạn rõ ràng và dễ đọc hơn, mà còn giúp tối ưu hóa hiệu suất và sử dụng bộ nhớ hiệu quả. Ví dụ, nếu bạn chỉ cần lưu trữ các số nguyên nhỏ, việc sử dụng Byte hoặc Short thay vì Int sẽ giúp tiết kiệm bộ nhớ.

**Kiểm tra null**

Kotlin cung cấp một số cách để kiểm tra null một cách an toàn:

- **Safe call operator ?.**: Cho phép bạn truy cập một đối tượng chỉ khi đối tượng đó không phải là null.
- **Elvis operator ?:**: Cung cấp một giá trị mặc định khi đối tượng là null.
- **Not-null assertion operator !!**: Khẳng định rằng đối tượng không phải là null.

**Ví dụ:**

```
val ten: String? = "Kotlin"

val doDai: Int? = ten?.length // doDai sẽ là 5

val doDaiMacDinh: Int = ten?.length ?: 0 // doDaiMacDinh sẽ là 5

val doDaiKhongAnToan: Int = ten!!.length // Có thể gây ra NPE nếu ten là null
```

**Ép Kiểu Dữ Liệu trong Kotlin**

Ép kiểu dữ liệu là quá trình chuyển đổi một kiểu dữ liệu thành kiểu dữ liệu khác. Kotlin cung cấp nhiều cách để thực hiện việc này, bao gồm ép kiểu an toàn và ép kiểu không an toàn.

**1. Ép Kiểu Không An toàn (Unsafe Casting)**

Ép kiểu không an toàn sử dụng **toán tử as**. Nếu việc ép kiểu không thành công, Kotlin sẽ ném ra một ngoại lệ **ClassCastException.**

**Ví dụ:**

```
val number: Int = 10

val string: String = number as String  // Sẽ ném ClassCastException
```

**2. Ép Kiểu An toàn (Safe Casting)**

Ép kiểu an toàn sử dụng toán tử **as?.** Nếu việc ép kiểu không thành công, kết quả sẽ là null.

**Ví dụ:**

```
val number: Int = 10

val string: String? = number as? String  // Kết quả là null
```

**3. Ép Kiểu giữa Các Kiểu Số**

Kotlin cung cấp các hàm chuyển đổi giữa các kiểu số như **toInt(), toLong(), toFloat(), toDouble(), toByte(), toShort(), và toChar()**

**Ví dụ:**

```
val number: Int = 10

val longNumber: Long = number.toLong()

val floatNumber: Float = number.toFloat()

val doubleNumber: Double = number.toDouble()

val byteNumber: Byte = number.toByte()
```

```
val shortNumber: Short = number.toShort()

val charNumber: Char = number.toChar()
```

**4. Ép Kiểu giữa Chuỗi và Số**

Kotlin cung cấp các hàm để chuyển đổi giữa chuỗi và số.

**Ví dụ:**

```
val stringNumber: String = "10"

val intNumber: Int = stringNumber.toInt()

val longNumber: Long = stringNumber.toLong()

val floatNumber: Float = stringNumber.toFloat()

val doubleNumber: Double = stringNumber.toDouble()
```

**=> Lưu ý:** Nếu chuỗi không thể chuyển đổi thành số, Kotlin sẽ ném ra một ngoại lệ **NumberFormatException.**

**Ví dụ:**

```
val invalidString: String = "abc"

val intNumber: Int = invalidString.toInt()  // Sẽ ném NumberFormatException
```

=> Để tránh ngoại lệ, bạn có thể sử dụng hàm **toIntOrNull(), toLongOrNull(), toFloatOrNull(), và toDoubleOrNull()**, các hàm này sẽ trả về **null** nếu chuỗi không thể chuyển đổi.

**Ví dụ:**

```
val invalidString: String = "abc"

val intNumber: Int? = invalidString.toIntOrNull()  // Kết quả là null
```

**5. Ép Kiểu giữa Chuỗi và Boolean**

Kotlin cung cấp các hàm **toBoolean()** và **toBooleanStrict()** để chuyển đổi chuỗi thành kiểu Boolean.

**Ví dụ:**

```
val trueString: String = "true"

val falseString: String = "false"

val booleanTrue: Boolean = trueString.toBoolean()  // Kết quả là true

val booleanFalse: Boolean = falseString.toBoolean()  // Kết quả là false
```

**=> Lưu ý:**

+ toBoolean() sẽ trả về true cho các chuỗi "true", "on", "yes", và "1", và false cho các chuỗi khác.

+ toBooleanStrict() chỉ trả về true cho "true" và false cho "false", nếu không sẽ ném ra một ngoại lệ **IllegalArgumentException**.

**Ví dụ:**

val invalidString: String = "on"

val booleanTrue: Boolean = invalidString.toBooleanStrict()  // Sẽ ném **IllegalArgumentException**

**6. Ép Kiểu giữa Các Kiểu Dữ Liệu Khác**

Kotlin hỗ trợ việc ép kiểu giữa các kiểu dữ liệu khác nhau, ví dụ như giữa Char và Int

**Ví dụ:**

val char: Char = 'A'

val int: Int = char.toInt()

=> Ép kiểu dữ liệu là một phần quan trọng trong lập trình, giúp bạn chuyển đổi giữa các kiểu dữ liệu một cách linh hoạt. Kotlin cung cấp các cách an toàn và không an toàn để thực hiện việc này, giúp bạn kiểm soát tốt hơn việc xử lý dữ liệu trong ứng dụng của mình.

**Các Toán Tử trong Kotlin**

Trong Kotlin, các toán tử là những ký hiệu đặc biệt được sử dụng để thực hiện các thao tác như tính toán, gán giá trị, so sánh, và logic. Kotlin cung cấp khả năng override các toán tử, cho phép bạn sử dụng các toán tử một cách linh hoạt và mạnh mẽ. (Giống như Java)

**1. Toán tử một ngôi**

Đây là loại toán tử chỉ tác động lên một toán hạng duy nhất. Kotlin có các toán tử một ngôi sau:

- **+a**: trả về giá trị của a.
- **-a**: trả về giá trị đối của a.
- **!a**: phủ định giá trị logic của a (true thành false, false thành true).
- **++a**: tăng giá trị của a lên 1 rồi trả về giá trị mới (tiền tố).
- **a++**: trả về giá trị của a rồi mới tăng giá trị lên 1 (hậu tố).
- **--a**: giảm giá trị của a xuống 1 rồi trả về giá trị mới (tiền tố).

- **a--**: trả về giá trị của a rồi mới giảm giá trị xuống 1 (hậu tố).

**Ví dụ:**

```
var a = 5
var b = -a // b = -5
var c = !true // c = false
var d = ++a // d = 6, a = 6
var e = a++ // e = 6, a = 7
```

**2. Toán tử số học cơ bản**

Dùng để thực hiện các phép toán số học cơ bản.

- +: cộng
- -: trừ
- *: nhân
- /: chia
- **%**: chia lấy dư

**Ví dụ:**

```
var a = 10 + 5 // a = 15
var b = 10 - 5 // b = 5
var c = 10 * 5 // c = 50
var d = 10 / 5 // d = 2
var e = 10 % 3 // e = 1
```

**3. Toán tử gán**

Dùng để gán giá trị cho biến.

- =: gán giá trị
- +=: cộng rồi gán (a += b tương đương a = a + b)
- -=: trừ rồi gán
- *=: nhân rồi gán
- /=: chia rồi gán
- **%=**: chia lấy dư rồi gán

**Ví dụ:**

```
var a = 10
```

a += 5 // a = 15

a -= 3 // a = 12

**4. Toán tử so sánh**

Dùng để so sánh hai giá trị.

- ==: bằng
- !=: khác
- >: lớn hơn
- <: nhỏ hơn
- >=: lớn hơn hoặc bằng
- <=: nhỏ hơn hoặc bằng

**Ví dụ:**

var a = 5 == 5 // a = true

var b = 5 != 3 // b = true

var c = 10 > 5 // c = true

**5. Toán tử logic**

Dùng để kết hợp các biểu thức logic.

- &&: và (AND)
- ||: hoặc (OR)
- !: phủ định (NOT)

**Ví dụ:**

var a = true && false // a = false

var b = true || false // b = true

var c = !true // c = false

**6. Độ ưu tiên toán tử**

Kotlin có quy tắc về độ ưu tiên của các toán tử. Các toán tử có độ ưu tiên cao hơn sẽ được thực hiện trước.

Ví dụ: 10 + 5 * 2 sẽ được tính là 10 + (5 * 2) = 20 chứ không phải (10 + 5) * 2 = 30.

**Lưu ý:**

Trong Kotlin, các toán tử thường được "overload" (nạp chồng) thành các phương thức. Điều này có nghĩa là bạn có thể sử dụng toán tử như các hàm thông thường.

Ví dụ: a + b tương đương với a.plus(b).

=> Các toán tử trong Kotlin là những công cụ quan trọng giúp bạn tạo ra các biểu thức và thực hiện các thao tác cần thiết trong mã lệnh. Việc hiểu rõ về các toán tử và cách sử dụng chúng sẽ giúp bạn viết mã hiệu quả và dễ đọc hơn.

**Nhập Dữ Liệu từ Bàn Phím với Kotlin**

**1. Giới Thiệu về Hàm readLine()**

Trong Kotlin, để nhập dữ liệu từ bàn phím, ta sử dụng hàm **readLine()**. Hàm này nằm trong thư viện mặc định **kotlin.io** và không cần import thêm bất kỳ thư viện nào khác.

- **Hàm readLine()**: Hàm này đọc một dòng dữ liệu từ bàn phím và trả về một chuỗi (String). Nếu không có dữ liệu nhập vào, hàm sẽ trả về null.

**2. Cách Sử Dụng readLine()**

Dưới đây là cách sử dụng hàm readLine() để nhập dữ liệu từ bàn phím:

```
fun main() {
    // Đọc một dòng dữ liệu từ bàn phím
    val input: String? = readLine()

    // Kiểm tra nếu input không phải null
    if (input != null) {
        println("Bạn đã nhập: $input")
    } else {
        println("Không có dữ liệu nhập vào.")
    }
}
```

**3. Ép Kiểu Dữ Liệu**

Đôi khi, bạn cần chuyển đổi dữ liệu nhập vào từ chuỗi sang các kiểu dữ liệu khác như Int, Double, Boolean, v.v. Kotlin cung cấp các hàm để thực hiện việc này.

**3.1. Ép Kiểu về Kiểu Số**

- **Chuyển đổi thành Int:**

```
fun main() {
    print("Nhập một số nguyên: ")
```

```
    val input: String? = readLine()
    if (input != null) {
        val number: Int? = input.toIntOrNull()
        if (number != null) {
            println("Số nguyên bạn đã nhập: $number")
        } else {
            println("Nhập không hợp lệ, vui lòng nhập một số nguyên.")
        }
    } else {
        println("Không có dữ liệu nhập vào.")
    }
}
```

- **Chuyển đổi thành Double:**

```
fun main() {
    print("Nhập một số thực: ")
    val input: String? = readLine()
    if (input != null) {
        val number: Double? = input.toDoubleOrNull()
        if (number != null) {
            println("Số thực bạn đã nhập: $number")
        } else {
            println("Nhập không hợp lệ, vui lòng nhập một số thực.")
        }
    } else {
        println("Không có dữ liệu nhập vào.")
    }
}
```

### 3.2. Ép Kiểu về Kiểu Boolean

- **Chuyển đổi thành Boolean:**

```
fun main() {
```

```
    print("Nhập true hoặc false: ")
    val input: String? = readLine()
    if (input != null) {
        val booleanValue: Boolean? = input.toBooleanStrictOrNull()
        if (booleanValue != null) {
            println("Giá trị boolean bạn đã nhập: $booleanValue")
        } else {
            println("Nhập không hợp lệ, vui lòng nhập true hoặc false.")
        }
    } else {
        println("Không có dữ liệu nhập vào.")
    }
}
```

### 3.3. Ép Kiểu về Kiểu Khác

- **Chuyển đổi thành Char:**

```
fun main() {
    print("Nhập một ký tự: ")
    val input: String? = readLine()
    if (input != null) {
        if (input.length == 1) {
            val charValue: Char = input[0]
            println("Ký tự bạn đã nhập: $charValue")
        } else {
            println("Nhập không hợp lệ, vui lòng nhập một ký tự.")
        }
    } else {
        println("Không có dữ liệu nhập vào.")
    }
}
```

### 4. Ví Dụ Minh Họa

```
fun main() {
    // Nhập tên
    print("Nhập tên của bạn: ")
    val name: String? = readLine()
    // Nhập tuổi
    print("Nhập tuổi của bạn: ")
    val ageInput: String? = readLine()
    // Nhập chiều cao
    print("Nhập chiều cao của bạn (cm): ")
    val heightInput: String? = readLine()

    // Nhập giới tính
    print("Nhập giới tính (true/false): ")
    val genderInput: String? = readLine()
    // Kiểm tra và xử lý dữ liệu
    if (name != null && ageInput != null && heightInput != null && genderInput != null) {
        val age: Int? = ageInput.toIntOrNull()
        val height: Double? = heightInput.toDoubleOrNull()
        val gender: Boolean? = genderInput.toBooleanStrictOrNull()

        if (age != null && height != null && gender != null) {
            println("Thông tin của bạn:")
            println("Tên: $name")
            println("Tuổi: $age")
            println("Chiều cao: $height cm")
            println("Giới tính: ${if (gender) "Nam" else "Nữ"}")
        } else {
            println("Nhập không hợp lệ, vui lòng kiểm tra lại các giá trị nhập vào.")
        }
```

```
    } else {
        println("Không có dữ liệu nhập vào.")
    }
}
```

=> Nhập dữ liệu từ bàn phím là một kỹ năng cơ bản nhưng quan trọng trong lập trình. Với Kotlin, hàm readLine() giúp bạn dễ dàng thực hiện việc này.