BÁO CÁO LAB 1.2 – Kotlin Basics

Họ và tên: Nguyễn Trung Kiên

MSSV: 20226110

Mã lớp: 161624

Môn học: Phát triển ứng dụng cho thiết bị di động - IT4785

Mã nguồn: https://github.com/trungkienit25/IT4785-HUST

1. Mục tiêu lab và thu hoạch đã đạt được.

- Cú pháp cơ bản của Kotlin.
- Thực hành các phép toán số học, ép kiểu, và khai báo biến.
- Điều kiện if/else, when với nhiều dạng khác nhau.
- Hiểu cơ chế null-safety (?. , ?: , !!).
- Làm việc với array, list và các vòng lặp (for, while, do…while, repeat).

2. Nội dung và kết quả thực hiện

2.1. Toán tử và Kiểu dữ liệu

- Thực hiện các phép toán số học cơ bản (+, -, *, /) trên số nguyên và số thực.

```
fun Lesson1p2_OperatorsAndTypes() {  1 Usage  new *
    println("\n## Phần 2: Toán tử và Kiểu dữ liệu")
    println("1 + 1 = ${1 + 1}")
    println("53 - 3 = ${53 - 3}")
    println("50 / 10 = ${50 / 10}")
    println("1.0 / 2.0 = ${1.0 / 2.0}")
    println("6.0 * 50 = ${6.0 * 50}")
    println("--------------------------------")
```

```
## Phần 2: Toán tử và Kiểu dữ liệu
1 + 1 = 2
53 - 3 = 50
50 / 10 = 5
1.0 / 2.0 = 0.5
6.0 * 50 = 300.0
----------------------------------
```

- Gọi phương thức trực tiếp trên số (times(), plus(), div()).

```
// Gọi phương thức trên số
println("2.times(3) = ${2.times( other = 3)}")
println("3.5.plus(4) = ${3.5.plus( other = 4)}")
println("2.4.div(2) = ${2.4.div( other = 2)}")
println("----------------------------------")
```

```
----------------------------------
2.times(3) = 6
3.5.plus(4) = 7.5
2.4.div(2) = 1.2
----------------------------------
```

- Thực hành ép kiểu tường minh (toByte(), toInt(), toDouble()).

```
// Ép kiểu
val i: Int = 6
val b1 = i.toByte()
println("Ép kiểu Int ($i) sang Byte: $b1")
val b2: Byte = 1
// val i1: Int = b2 // Lỗi: không thể gán Byte cho Int
val i4: Int = b2.toInt() // OK: Phải ép kiểu tường minh
println("Ép kiểu Byte ($b2) sang Int: $i4")
println("---------------------------------")
```

```
---------------------------------
Ép kiểu Int (6) sang Byte: 6
Ép kiểu Byte (1) sang Int: 1
---------------------------------
```

- So sánh var (có thể thay đổi giá trị) và val (bất biến).

```
// Biến var và val
var fish = 1
fish = 2
val aquarium = 1
println("Biến var 'fish' đã thay đổi giá trị: $fish")
println("Biến val 'aquarium' là bất biến: $aquarium")
println("---------------------------------")
```

```
38      // Biến var và val
39      var fish = 1
40      fish = 2
41      val aquarium = 1
42      aquarium = 1
43      println("Biến var 'fish' đã thay đổi giá trị: $fish")
44      println("Biến val 'aquarium' là bất biến: $aquarium")
45      println("-----------------------------------")
46
47      // String templates

C:\CodeLibrary\HelloKotlin\src\Main.kt:42:5
Kotlin: 'val' cannot be reassigned.
```

```
-----------------------------------
Biến var 'fish' đã thay đổi giá trị: 2
Biến val 'aquarium' là bất biến: 1
-----------------------------------
```

- Sử dụng string template để chèn biến và biểu thức vào chuỗi.

```
    // String templates
    val numberOfFish = 5
    val numberOfPlants = 12
    println("I have $numberOfFish fish and $numberOfPlants plants")
    println("I have ${numberOfFish + numberOfPlants} total items")
}
```

```
-----------------------------------
I have 5 fish and 12 plants
I have 17 total items
```

2.2. Điều kiện và Booleans

Kết quả chung của 4 cases sau khi chạy:

```
## Phần 3: Điều kiện và Booleans
Case 1 - if/else: Good ratio!
Case 2 - if với range: fish (50) is in range 1..100
Case 3 - if/else if/else: That's a lot of fish!
Case 4 - when: That's a lot of fish!
```

- Viết câu lệnh if/else so sánh số cá và số cây.

```
fun Lesson1p3_Conditions() {  1 Usage  new *
    println("\n## Phần 3: Điều kiện và Booleans")
    // Case 1: if/else cơ bản
    val numberOfFish = 50
    val numberOfPlants = 23
    print("Case 1 - if/else: ")
    if (numberOfFish > numberOfPlants) {
        println("Good ratio!")
    } else {
        println("Unhealthy ratio")
    }
```

- Sử dụng range (ví dụ: 1..100) trong điều kiện.

```
// Case 2: if với range
val fish = 50
print("Case 2 - if với range: ")
if (fish in 1 ≤ .. ≤ 100) {
    println("fish ($fish) is in range 1..100")
}
```

- Viết cấu trúc if/else if/else để phân loại số lượng cá.

```
// Case 3: if/else if/else
print("Case 3 - if/else if/else: ")
if (numberOfFish == 0) {
    println("Empty tank")
} else if (numberOfFish < 40) {
    println("Got fish!")
} else {
    println("That's a lot of fish!")
}
```

- Dùng when thay thế cho switch trong các ngôn ngữ khác.

```
// Case 4: when statement
print("Case 4 - when: ")
when (numberOfFish) {
    0 -> println("Empty tank")
    in 1 ≤ .. ≤ 39 -> println("Got fish!")
    else -> println("That's a lot of fish!")
}
```

2.3. Nullability

- Kết quả cuối khi chạy:

```
## Phần 4: Nullability
Biến nullable 'marbles' có thể giữ giá trị: null
---------------------------------
Dùng safe call '?.' để giảm giá trị: 5
Dùng elvis operator '?:' khi biến là null: 0
```

- Biến kiểu nguyên thủy (Int) không thể gán null. Khai báo biến nullable bằng Int?.

```
fun Lesson1p4_Nullability() {  1 Usage  new *
    println("\n## Phần 4: Nullability")
    // var rocks: Int = null // Lỗi
    var marbles: Int? = null
    println("Biến nullable 'marbles' có thể giữ giá trị: $marbles")
    println("----------------------------------")
```

- Sử dụng safe call ?. để tránh lỗi NullPointerException.

```
    // Safe call ?.
    var fishFoodTreats: Int? = 6
    fishFoodTreats = fishFoodTreats?.dec()
    println("Dùng safe call '?.' để giảm giá trị: $fishFoodTreats")
```

- Dùng Elvis operator ?: để gán giá trị mặc định khi biến bị null.

```
// Elvis operator ?:
fishFoodTreats = null
val treats = fishFoodTreats?.dec() ?: 0
println("Dùng elvis operator '?:' khi biến là null: $treats")
```

2.4. Mảng, List và Vòng lặp

```
## Phần 5: Mảng, List, và Vòng lặp
listOf (bất biến): [mackerel, trout, halibut]
mutableListOf (có thể thay đổi): [tuna, salmon]
----------------------------------
Vòng lặp for:
shark salmon minnow
Vòng lặp for với index:
Item at 0 is shark
Item at 1 is salmon
Item at 2 is minnow
----------------------------------
Các loại vòng lặp for khác:
Range 1..5: 12345
Range 5 downTo 1: 54321
Range 3..6 step 2: 35
Range 'd'..'g': defg
```

- Tạo listOf (bất biến) và mutableListOf (có thể thay đổi).

```
fun Lesson1p5_ArraysLoops() {  1 Usage   new *
    println("\n## Phần 5: Mảng, List, và Vòng lặp")
    val school = listOf("mackerel", "trout", "halibut")
    println("listOf (bất biến): $school")

    val myList = mutableListOf("tuna", "salmon", "shark")
    myList.remove( element = "shark")
    println("mutableListOf (có thể thay đổi): $myList")
    println("----------------------------------")
```

- Thực hiện thao tác thêm, xóa phần tử trong list.

```
// Thao tác XÓA phần tử
myList.remove( element = "shark")
println("List sau khi xóa 'shark': $myList")

// Thao tác THÊM phần tử
myList.add("shark")
println("List sau khi thêm 'shark': $myList")
println("-----------------------------------")
```

- Khai báo mảng (arrayOf) và duyệt bằng vòng lặp for.

```
val schoolArray = arrayOf("shark", "salmon", "minnow")
println("Vòng lặp for:")
for (element in schoolArray) {
    print("$element ")
}
println()
```

- Duyệt mảng với cả chỉ số và giá trị (withIndex()).

```
println("Vòng lặp for với index:")
for ((index, element) in schoolArray.withIndex()) {
    println("Item at $index is $element")
}
println("-----------------------------------")
```

- Thử các dạng vòng lặp:
  - for (i in 1..5)
  - for (i in 5 downTo 1)
  - for (i in 3..6 step 2)
  - for (i in 'd'..'g')

```
println("Các loại vòng lặp for khác:")
print("Range 1..5: ")
for (i in 1 ≤ .. ≤ 5) print(i)
println()

print("Range 5 downTo 1: ")
for (i in 5 ≥  downTo  ≥ 1) print(i)
println()

print("Range 3..6 step 2: ")
for (i in 3 ≤ .. ≤ 6 step 2) print(i)
println()

print("Range 'd'..'g': ")
for (i in 'd' ≤ .. ≤ 'g') print (i)
println()
```