

## BÁO CÁO LAB 2

Họ và tên: Nguyễn Trung Kiên

MSSV: 20226110

Mã lớp: 161624

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

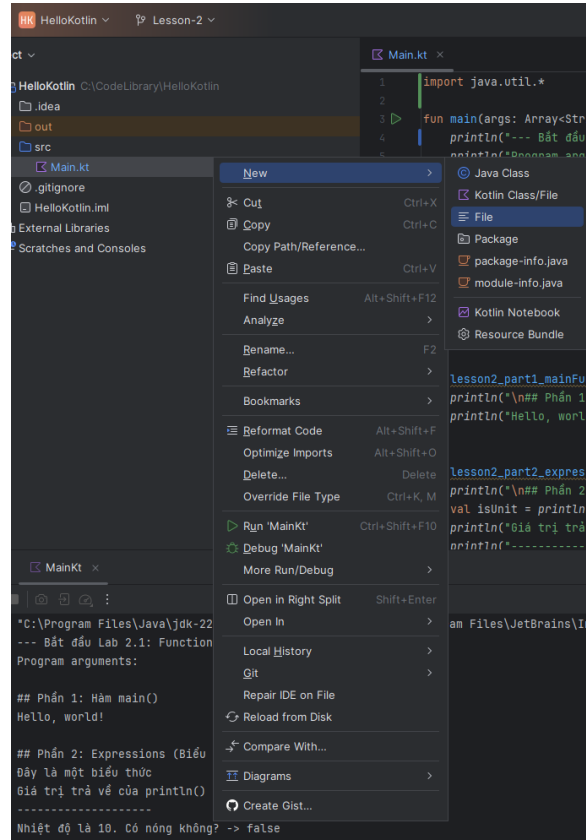
Mã nguồn: <https://github.com/trungkienit25/IT4785-HUST>

### Contents

1. Explore the main() function.....	2
1.1. Create a Kotlin file .....	2
1.2. Add code and run your program .....	2
1.3. Pass arguments to main().....	3
1.4. Change the code to use string template .....	4
2. Learn why (almost) everything has a value.....	5
3. Learn more about function .....	6
3.1. Create some functions .....	6
3.2. Use a when expression .....	6
4. Explore default values and compact functions .....	7
4.1. Create a default value for a parameter .....	7
4.2. Add required parameters.....	8
4.3. Make compact functions.....	9
5. Get started with filters .....	9
6. Get started with lambdas and higher-order functions.....	11
6.1. Learn about lambdas.....	11
6.2. Create a higher-order function.....	11

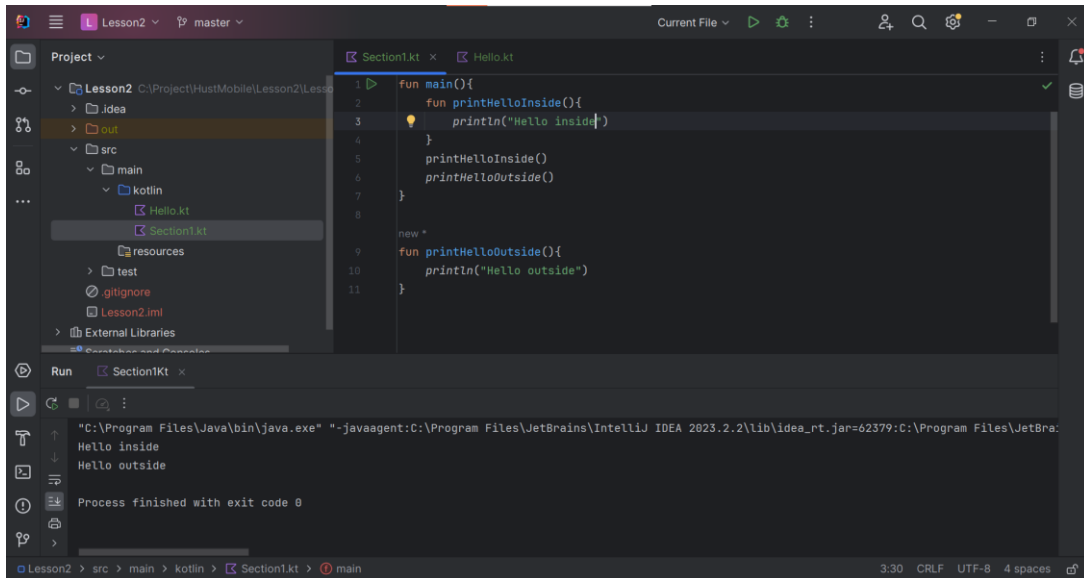
## 1. Explore the main() function

### 1.1. Create a Kotlin file



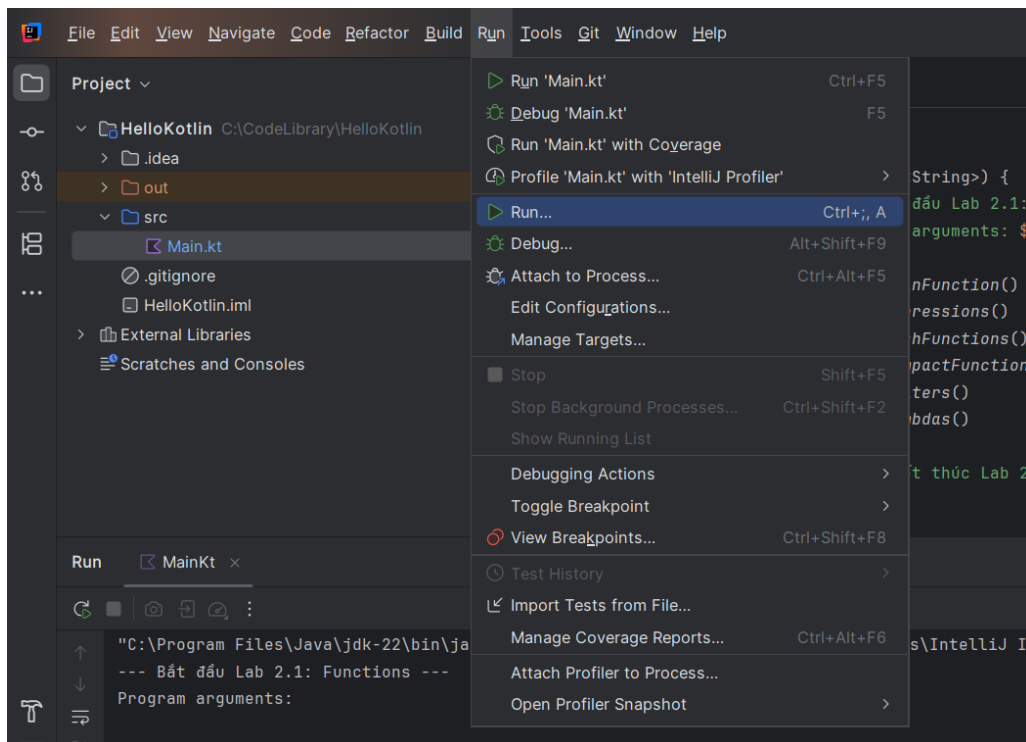
Các bước thực hiện: Chuột phải vào thư mục “kotlin” => Chọn New => Chọn File => Điền tên file: “[Tên file].kt” (ví dụ: vidu.kt) => Enter

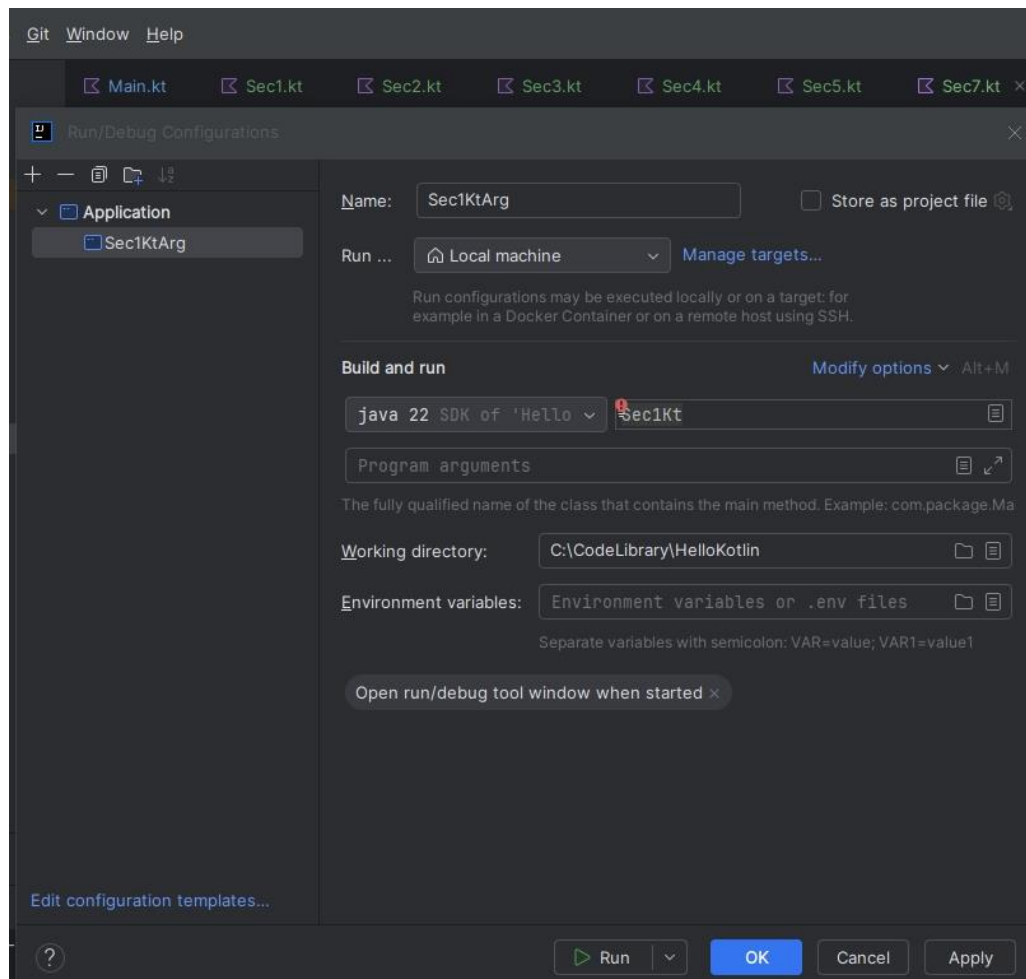
### 1.2. Add code and run your program



- Việc định nghĩa hàm có thể là bên ngoài hoặc bên trong 1 hàm.
- Ví dụ, hàm `printHelloInside()` được định nghĩa trong hàm `main()`, `printHelloOutside` được định nghĩa ngoài hàm `main()`.
- 2 hàm cùng được gọi trong `main()`, không gây ra lỗi kể cả khi 1 hàm được khai báo trong một hàm khác.

### 1.3. Pass arguments to main()

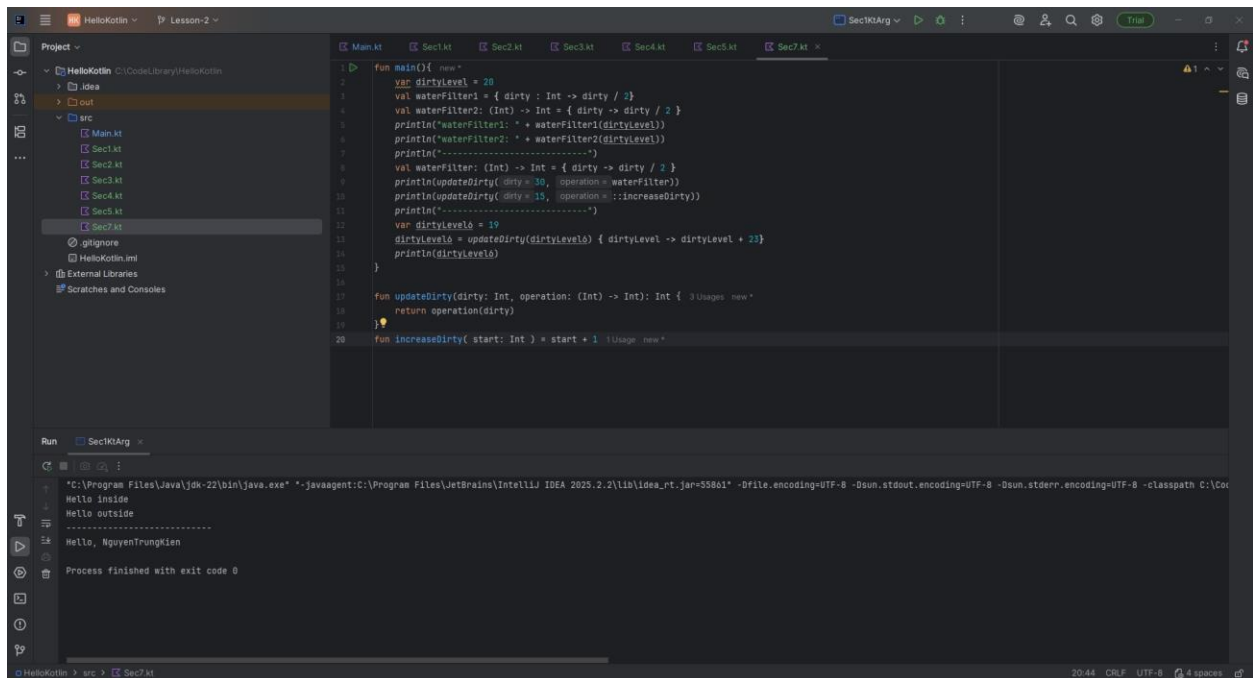




Bước thực hiện: Chọn Run (trên thanh toolbar) => Chọn Edit Configuration => Chọn “+” (góc trái trên của cửa sổ hiện ra) => Chọn Application => Điền thông tin => Ok

Mô tả: Dùng cách này ta có thể truyền tham số vào hàm main().

#### ***1.4. Change the code to use string template***

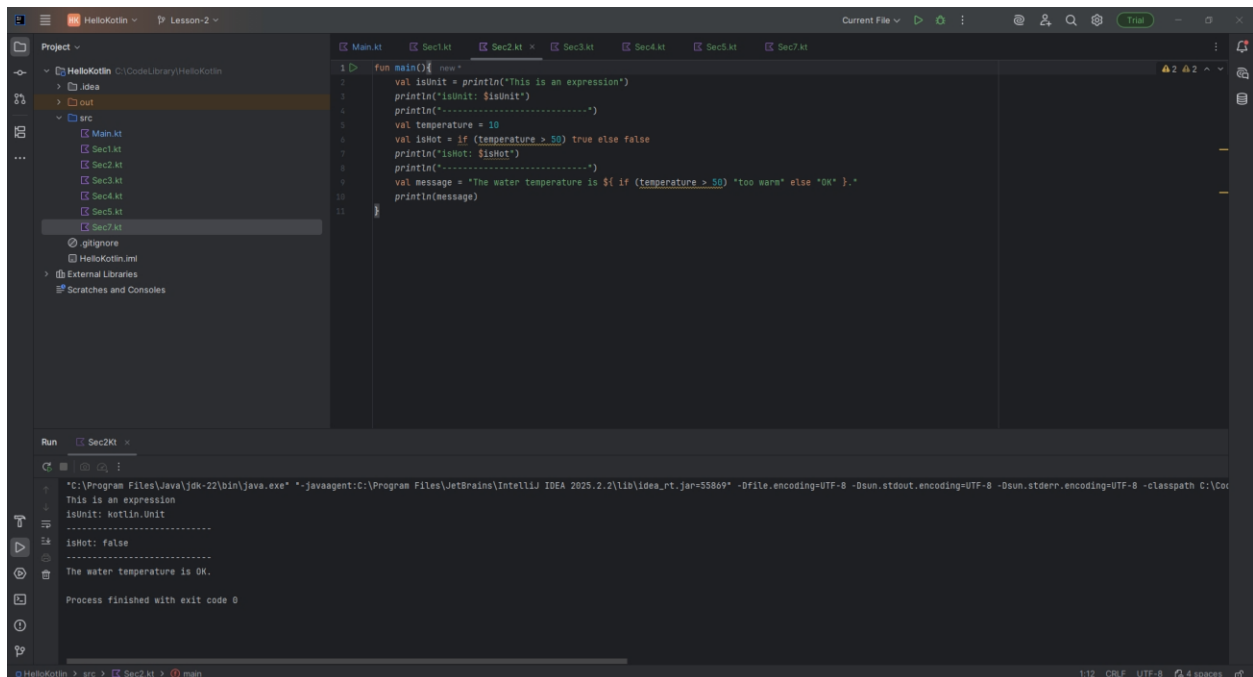


Bước thực hiện: Chuyển chương trình muốn chạy thành Application vừa thực hiện => Chạy chương trình

Mô tả:

Trong phần trên ta đã truyền vào trong biến mảng Args một phân tử dạng chuỗi có giá trị “NguyenTrungKien”

## 2. Learn why (almost) everything has a value

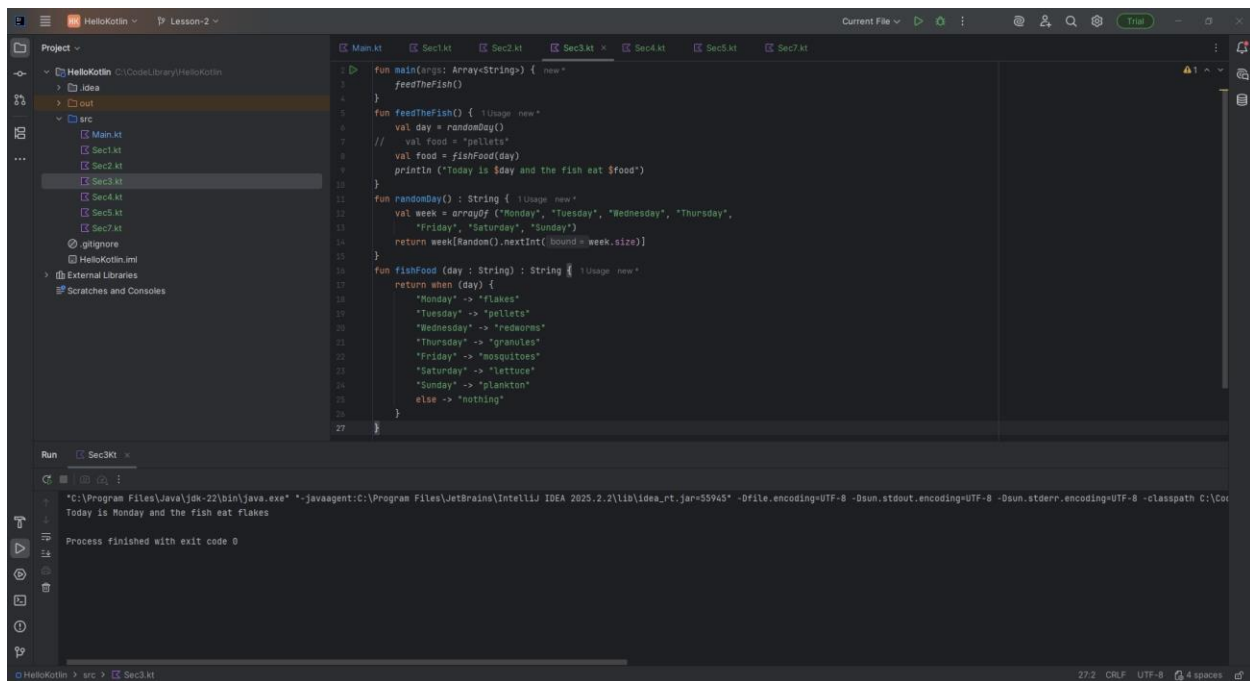


Mô tả:

- Hầu hết mọi phần tử trong kotlin đều có giá trị.
- Vòng lặp là trường hợp ngoại lệ. Không có giá trị nào cho for hoặc while.
- Nếu bạn cố gán giá trị của vòng lặp, trình biên dịch sẽ báo lỗi.

### 3. Learn more about function

#### 3.1. Create some functions



Mô tả:

- Khi hàm main() chạy, sẽ gọi hàm feedTheFish().
- Giá trị day sẽ được gọi từ kết quả của hàm randomDay().
- Hàm randomDay sử dụng hàm dựng sẵn là Random() trong thư viện java util, trả về là một trong các ngày đã được định nghĩa trong mảng week.
- Các giá trị của mảng này là chuỗi tên các ngày trong tuần.

#### 3.2. Use a when expression

```

new *
5 fun feedTheFish() {
6     val day = randomDay()
7     // val food = "pellets"
8     val food = fishFood(day)
9     println ("Today is $day and the fish eat $food")
10 }

```

```

C:\Program Files\Java\bin\java.exe" "-javaagent:C:\Program Files\JetBr
Today is Monday and the fish eat flakes

Process finished with exit code 0

```

Mô tả:

Thay vì cố định giá trị food, ta cài đặt thêm một hàm nhận vào giá trị day và trả về giá trị food tương ứng

```

16 fun fishFood (day : String) : String {
17     return when (day) {
18         "Monday" -> "flakes"
19         "Tuesday" -> "pellets"
20         "Wednesday" -> "redworms"
21         "Thursday" -> "granules"
22         "Friday" -> "mosquitoes"
23         "Saturday" -> "lettuce"
24         "Sunday" -> "plankton"
25         else -> "nothing"
26     }
27 }

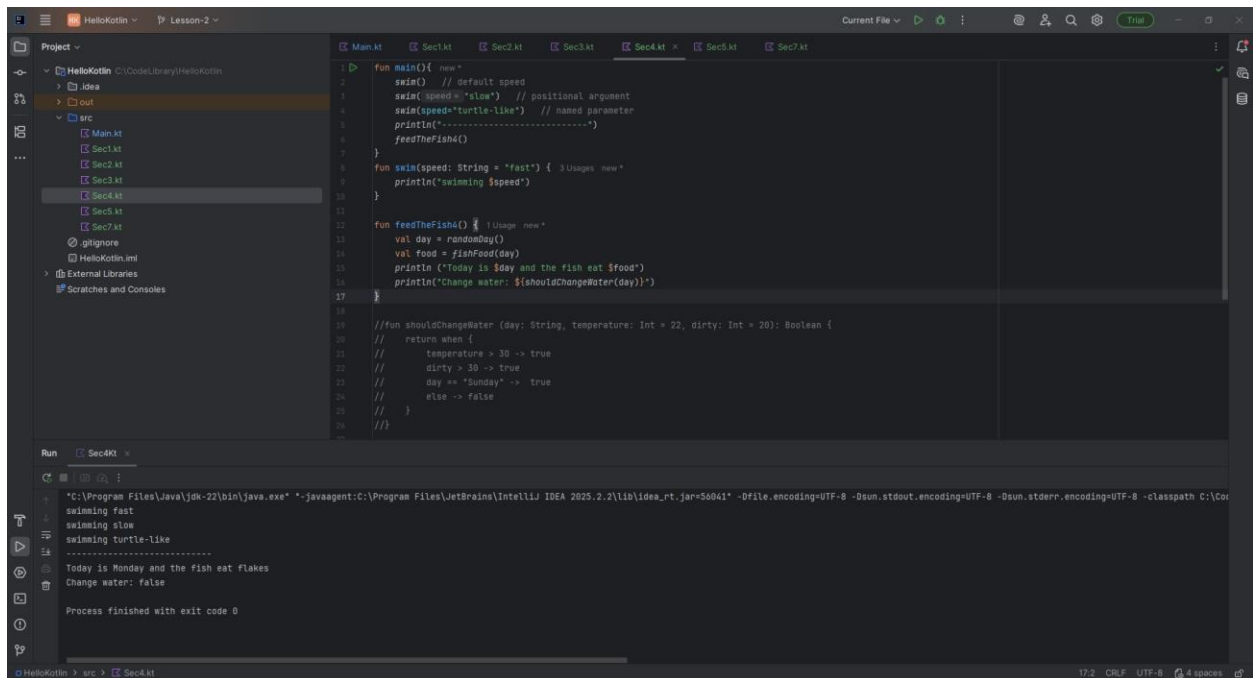
```

Mô tả:

- Ta có thể trực tiếp return về hàm kết quả của mệnh đề when.
- Trường hợp default là trả về nothing (ngoại lệ).

## 4. Explore default values and compact functions

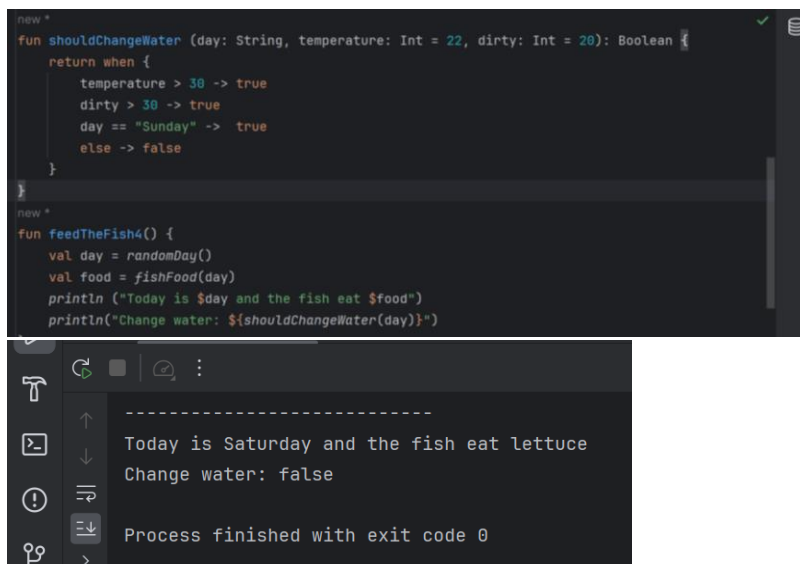
### 4.1. Create a default value for a parameter



Mô tả:

- Khi không truyền vào tham số, hàm sẽ nhận giá trị mặc định của tham số speed là “fast”.
- Khi chỉ truyền vào “slow”, hàm sẽ lấy theo thứ tự. Vì chỉ có mỗi speed là tham số nên hàm sẽ nhận giá trị “slow” cho tham số.
- Truyền speed = “turtle-like” là kiểu truyền tường minh, hàm nhiều tham số và truyền không đúng thứ tự theo khai báo, speed vẫn sẽ nhận được giá trị “turtle-like”.

## 4.2. Add required parameters





Mô tả:

- Trường hợp không đặt giá trị mặc định, khi gọi hàm bắt buộc phải truyền vào giá trị cho tham số đó.

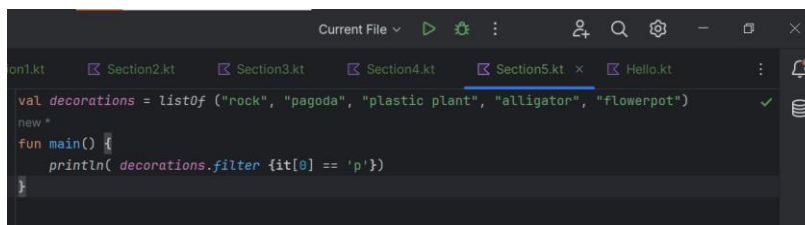
### 4.3. Make compact functions

```
28 fun shouldChangeWater (day: String, temperature: Int = 22, dirty: Int = 20): Boolean {
29     return when {
30         isTooHot(temperature) -> true
31         isDirty(dirty) -> true
32         isSunday(day) -> true
33         else -> false
34     }
35 }
new *
36 fun isTooHot(temperature: Int) = temperature > 30
new *
37 fun isDirty(dirty: Int) = dirty > 30
new *
38 fun isSunday(day: String) = day == "Sunday"
```

Mô tả: Cách viết ngắn gọn khi hàm là phép tính đơn giản.

## 5. Get started with filters

### 5.1. Create a filter



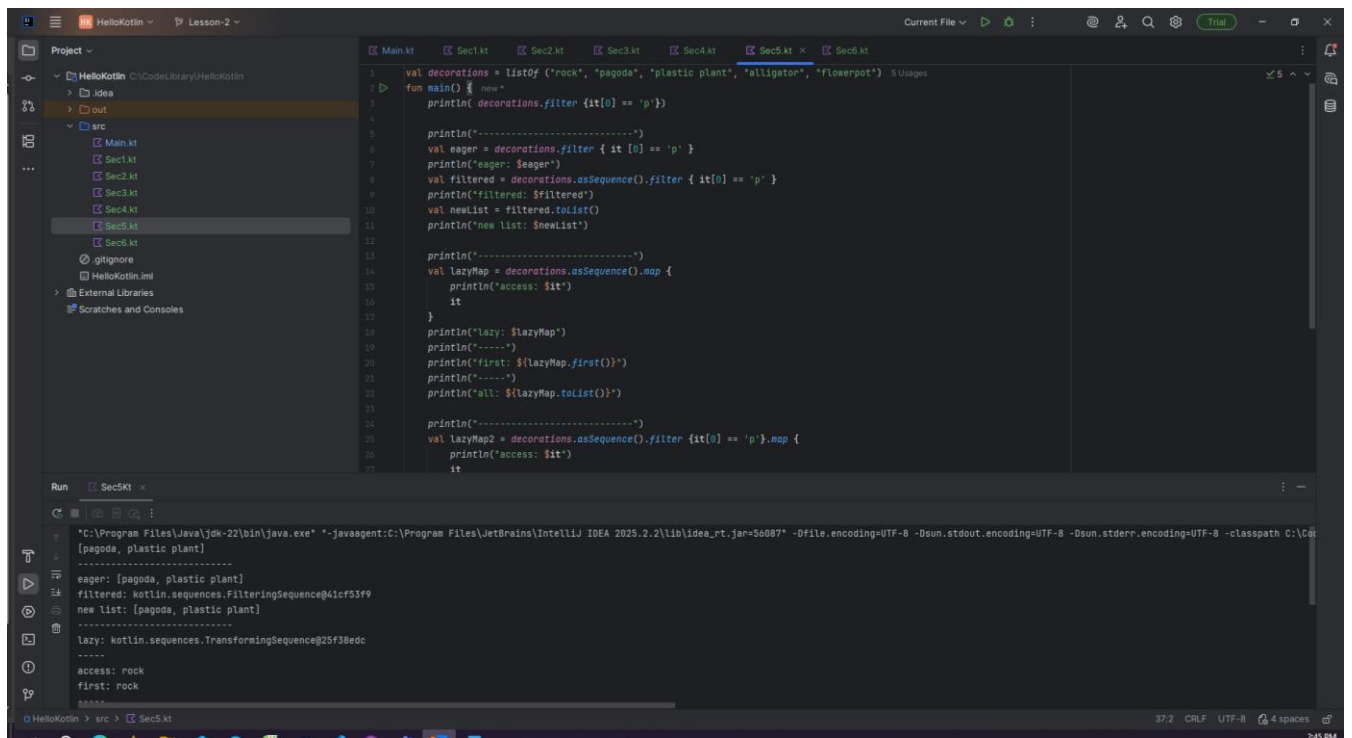
```
on1.kt | Section2.kt | Section3.kt | Section4.kt | Section5.kt x | Hello.kt
val decorations = listOf("rock", "pagoda", "plastic plant", "alligator", "flowerpot")
new *
fun main() {
    println(decorations.filter { it[0] == 'p' })
}
```

Mô tả:

- filter cho phép kiểm tra các phần tử trong một danh sách, it là tham chiếu của mỗi phần tử, it[0] trả về giá trị đầu tiên của phần tử.

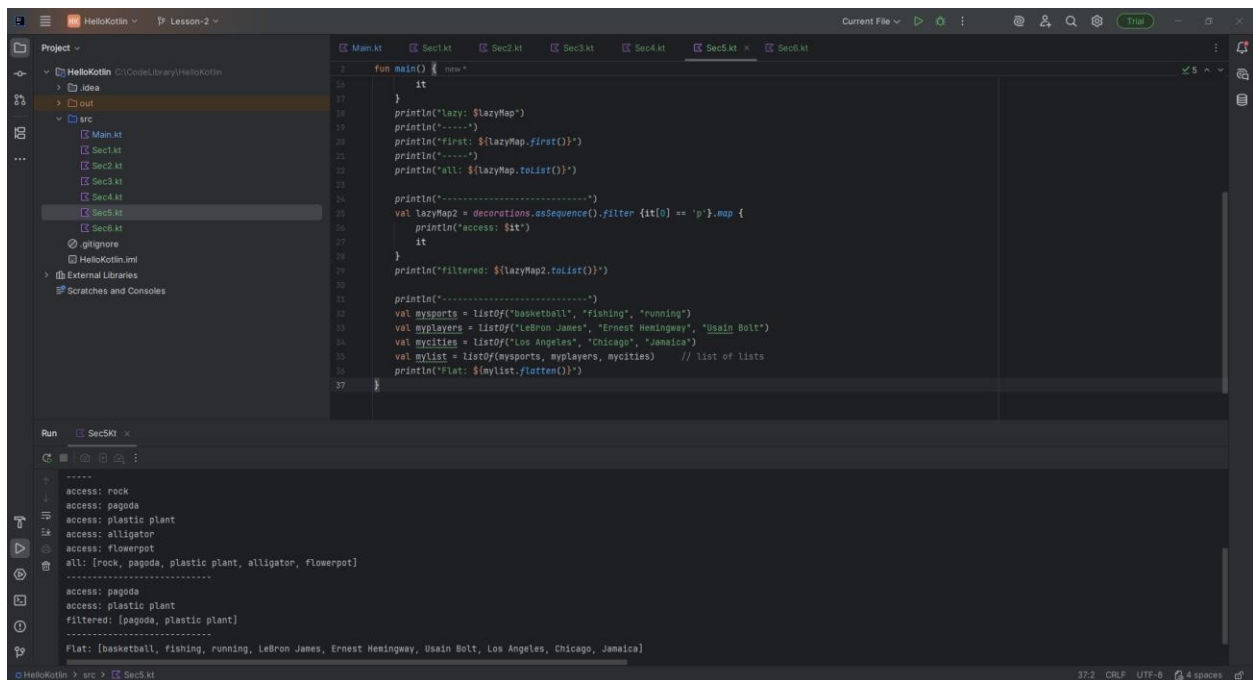
- Hàm trên sẽ trả về các phần tử bắt đầu bằng “p”.

### 5.2. Compare eager and lazy filters



Mô tả:

Eager sẽ load hết cùng lúc, Lazy thì sẽ load những gì cần thiết.

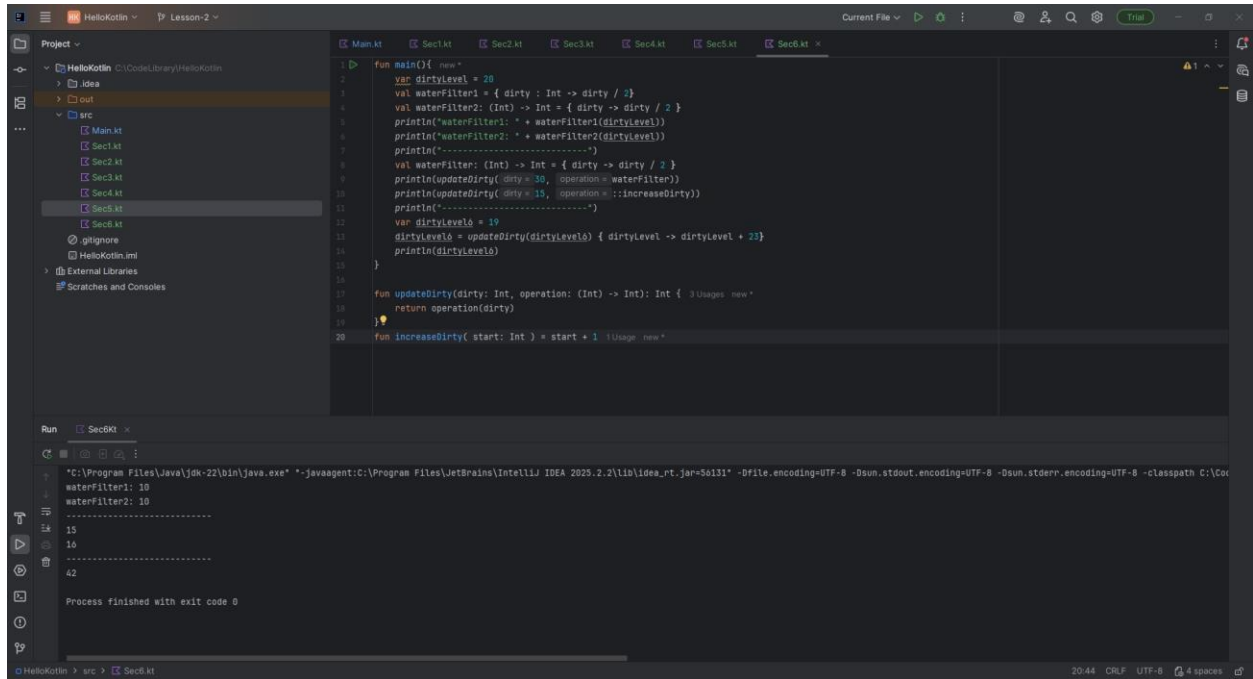


Mô tả:

Sử dụng toList() hoặc flatten() để hiển thị ra một danh sách.

## 6. Get started with lambdas and higher-order functions

### 6.1. Learn about lambdas



```
1 fun main() { new *
2     var dirtyLevel = 20
3     val waterFilter1 = { dirty: Int -> dirty / 2 }
4     val waterFilter2: (Int) -> Int = { dirty -> dirty / 2 }
5     println("waterFilter1: " + waterFilter1(dirtyLevel))
6     println("waterFilter2: " + waterFilter2(dirtyLevel))
7     println("-----")
8     val waterFilter: (Int) -> Int = { dirty -> dirty / 2 }
9     println(updateDirty(dirty=20, operation = waterFilter))
10    println(updateDirty(dirty=15, operation = ::increaseDirty))
11    println("-----")
12    var dirtyLevel = 19
13    dirtyLevel = updateDirty(dirtyLevel) { dirtyLevel -> dirtyLevel + 23 }
14    println(dirtyLevel)
15 }
16
17 fun updateDirty(dirty: Int, operation: (Int) -> Int): Int { 3 Usages new *
18     return operation(dirty)
19 }
20 fun increaseDirty( start: Int ) = start + 1 // Usage new *
```

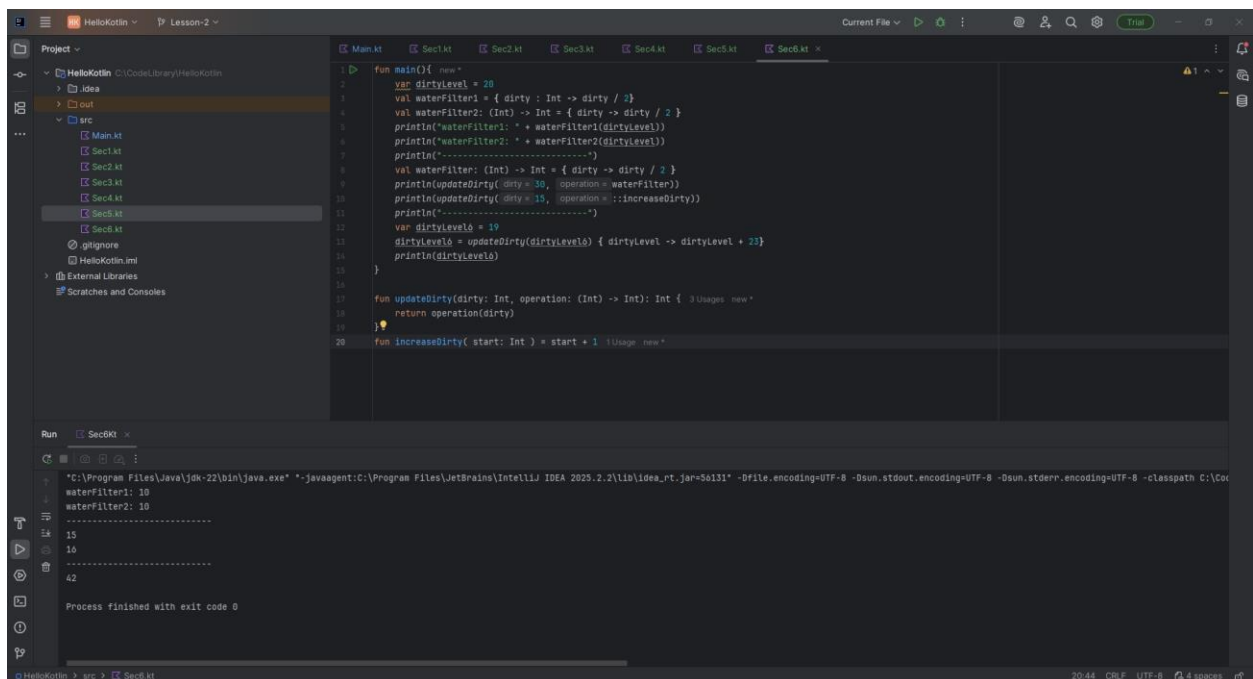
Run console output:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.2.2\lib\idea_rt.jar=50131" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath C:\Co
waterFilter1: 10
waterFilter2: 10
-----
15
16
-----
42
Process finished with exit code 0
```

Mô tả:

Biến có thể nhận giá trị là một hàm có thể truyền vào tham số

### 6.2. Create a higher-order function



```
1 fun main() { new *
2     var dirtyLevel = 20
3     val waterFilter1 = { dirty: Int -> dirty / 2 }
4     val waterFilter2: (Int) -> Int = { dirty -> dirty / 2 }
5     println("waterFilter1: " + waterFilter1(dirtyLevel))
6     println("waterFilter2: " + waterFilter2(dirtyLevel))
7     println("-----")
8     val waterFilter: (Int) -> Int = { dirty -> dirty / 2 }
9     println(updateDirty(dirty=20, operation = waterFilter))
10    println(updateDirty(dirty=15, operation = ::increaseDirty))
11    println("-----")
12    var dirtyLevel = 19
13    dirtyLevel = updateDirty(dirtyLevel) { dirtyLevel -> dirtyLevel + 23 }
14    println(dirtyLevel)
15 }
16
17 fun updateDirty(dirty: Int, operation: (Int) -> Int): Int { 3 Usages new *
18     return operation(dirty)
19 }
20 fun increaseDirty( start: Int ) = start + 1 // Usage new *
```

Run console output:

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.2.2\lib\idea_rt.jar=50131" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath C:\Co
waterFilter1: 10
waterFilter2: 10
-----
15
16
-----
42
Process finished with exit code 0
```

Mô tả:

- Hàm bậc cao: lấy các hàm khác làm tham số hoặc là một hàm trả về một hàm khác.
- Ta truyền lambda cho một hàm bậc cao lấy một hàm làm đối số.
- Để chỉ định đối số là một hàm thông thường, sử dụng toán tử ::.