

Họ tên: Lê Phúc Hưng

MSSV: 20215276

Mã lớp: 151902

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

Source Code: <https://github.com/lephuchung/HustMobile>

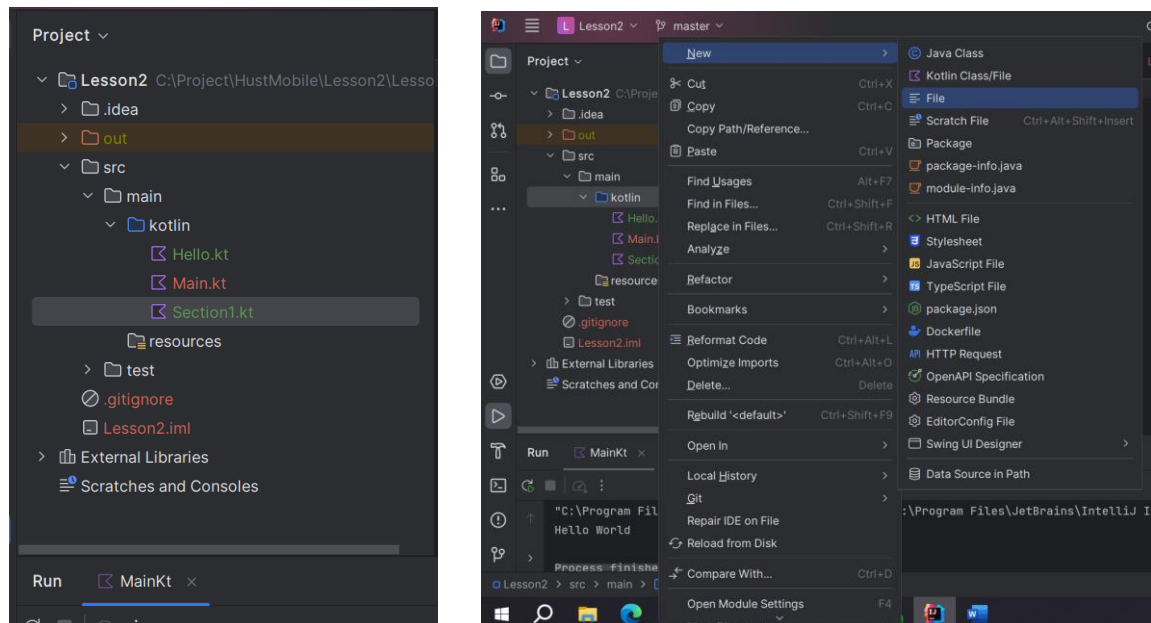
Bài: Lesson 2 - Function

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.....	4
3. Learn more about function	5
3.1. Create some functions	5
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.....	8
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.....	12

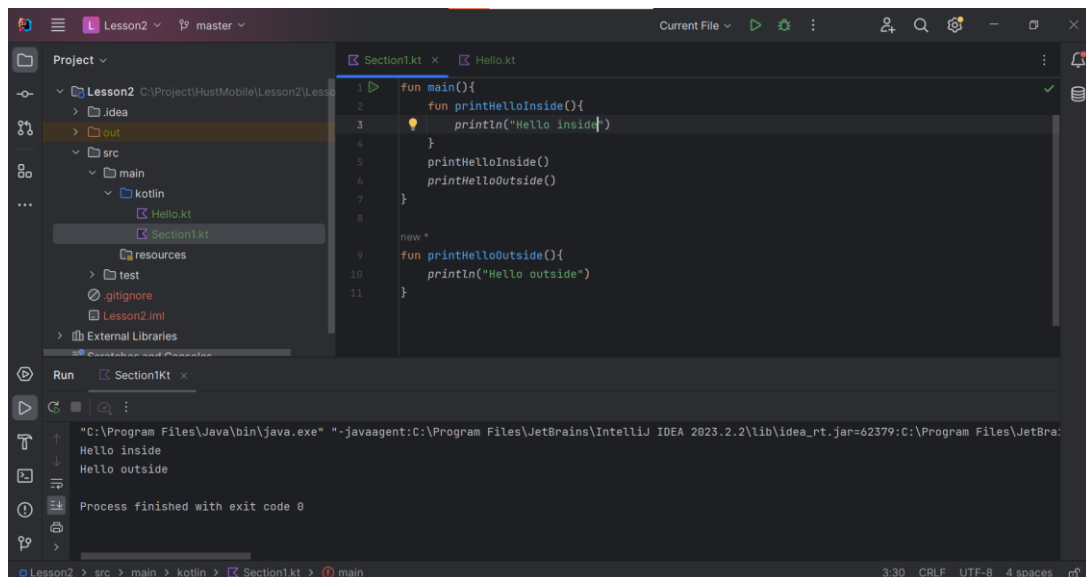
1. Explore the main() function

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

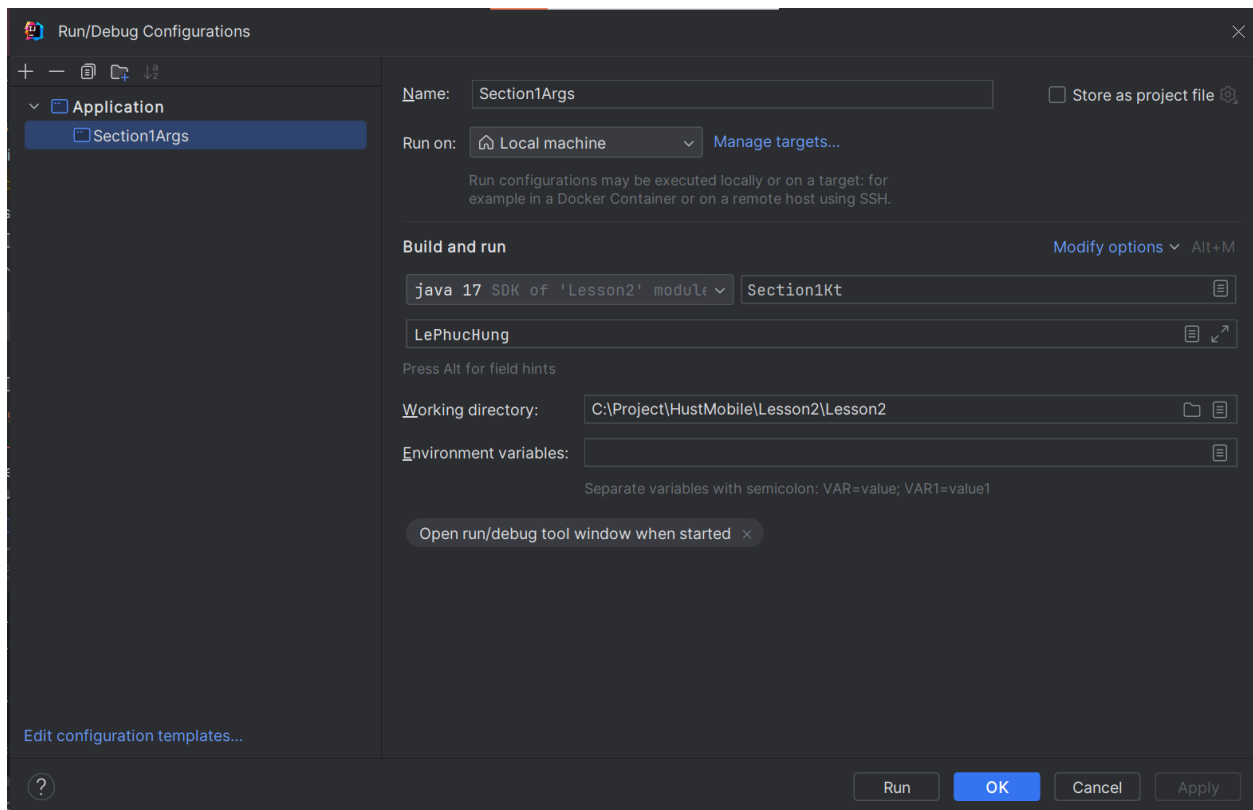
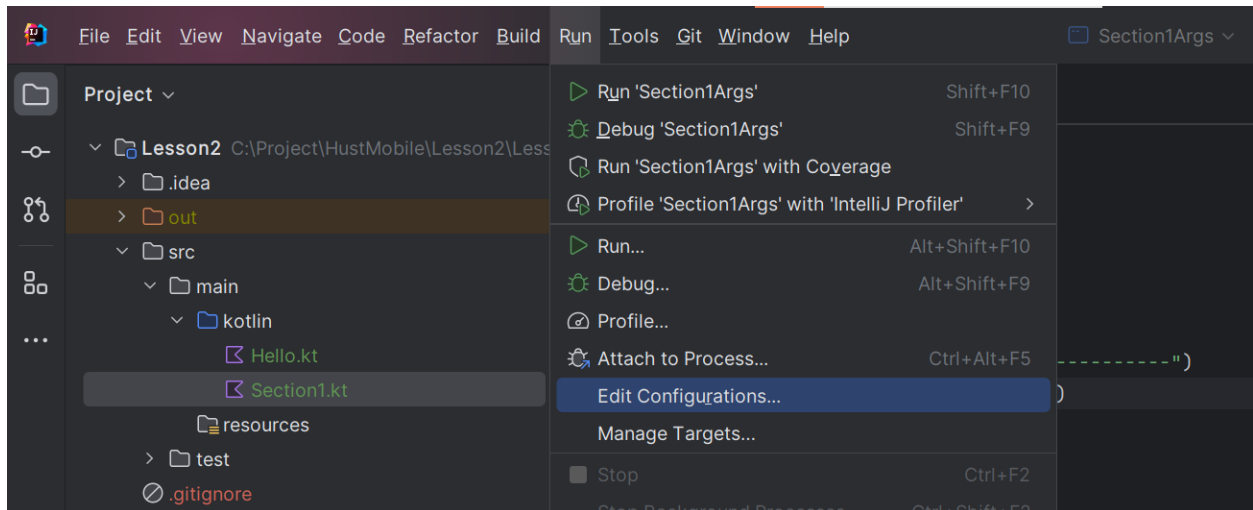
1.2. Add code and run your program



Mô tả:

Việc định nghĩa hàm có thể diễn ra bên ngoài hoặc bên trong 1 hàm. Như trên, hàm `printHelloInside()` được định nghĩa trong hàm `main()`, `printHelloOutside` được định nghĩa ngoài hàm `main()`. Sau đó 2 hàm cùng được gọi trong `main()`. Điều này không gây ra lỗi kể cả khi 1 hàm được khai báo trong một hàm khác không phải `main()`

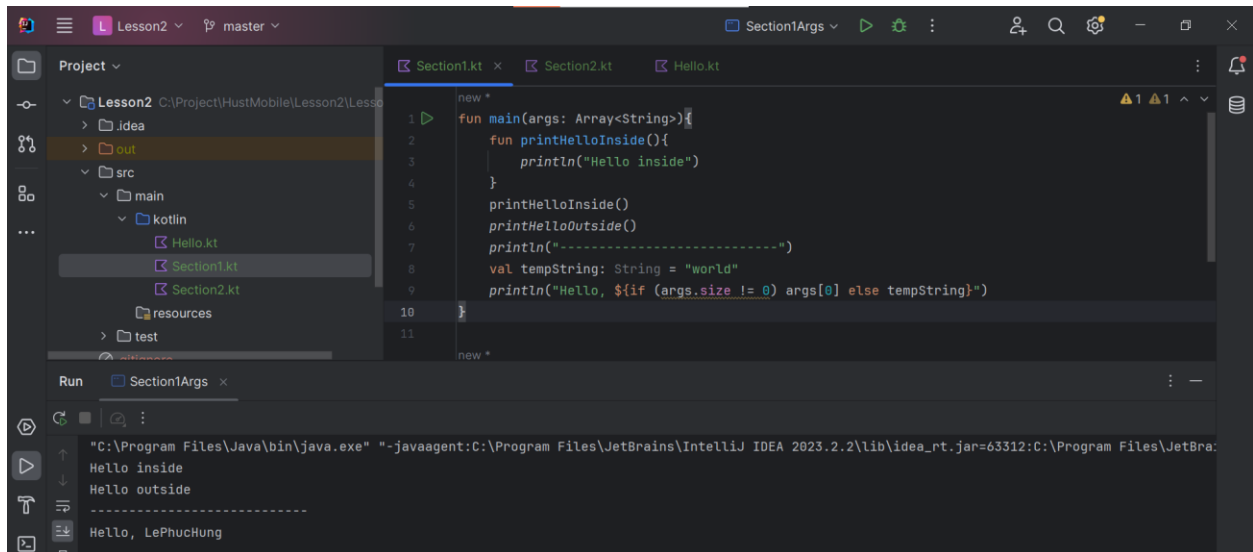
1.3. Pass arguments to main()



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

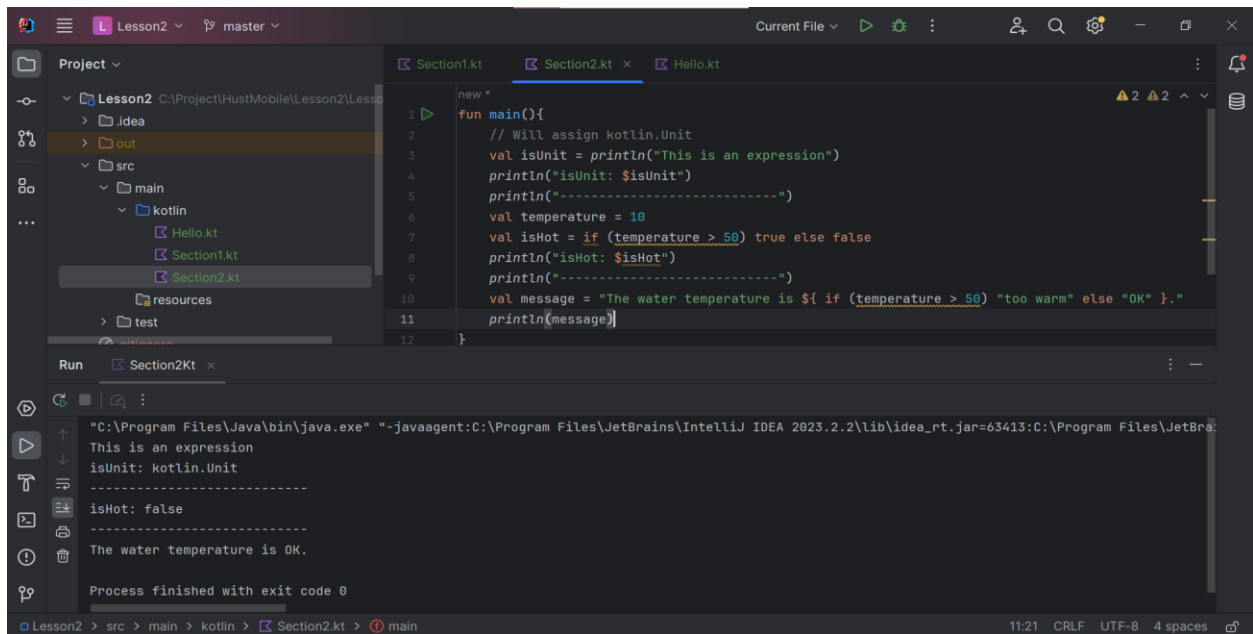


Bước thực hiện: Chuyển chương trình muốn chạy thành Application vừa thực hiện => Chạy chương trình

Mô tả:

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

2. Learn why (almost) everything has a value

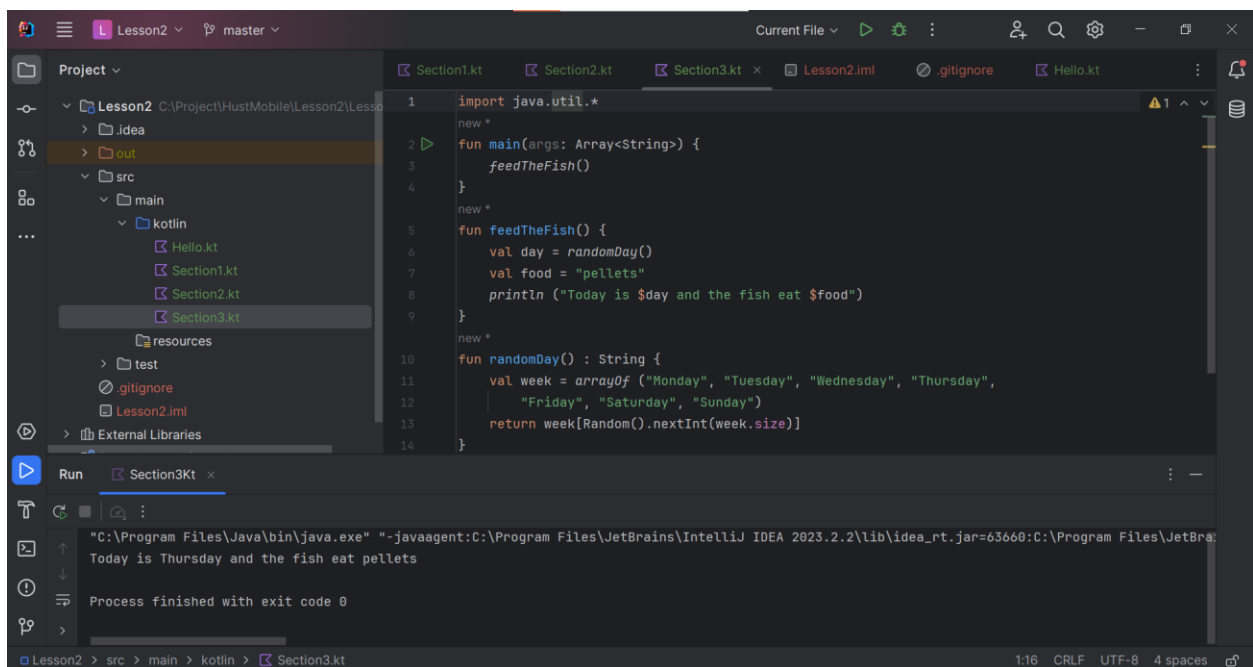


Mô tả:

Hầu hết mọi phân tử trong kotlin đều có giá trị. Vòng lặp là ngoại lệ của "mọi thứ đều có giá trị". Không có giá trị hợp lý nào cho vòng lặp for hoặc vòng lặp while, vì vậy chúng không có giá trị. Nếu bạn cố gán giá trị của vòng lặp cho một thứ gì đó, trình biên dịch sẽ báo lỗi.

3. Learn more about function

3.1. Create some functions



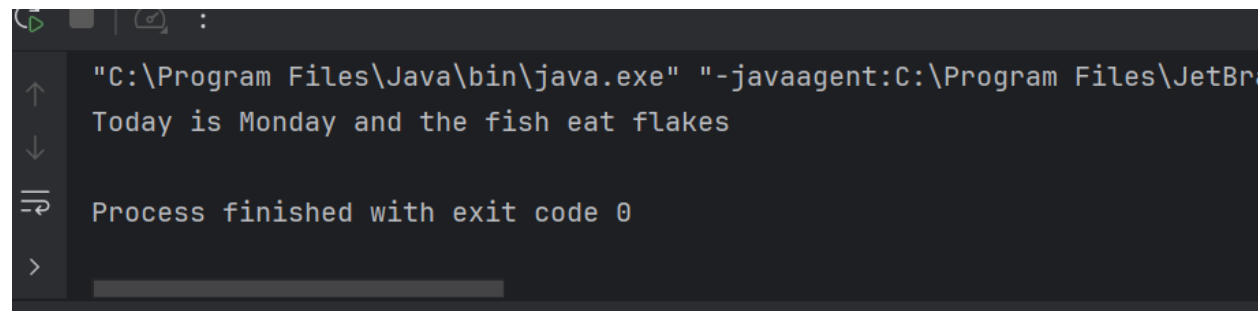
Mô tả:

Khi hàm main() chạy, hàm feedTheFish() sẽ được gọi. Khi hàm feedTheFish() chạy, 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 sẵn 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 }

16 fun fishFood (day : String) : String {
17     var food = ""
18     when (day) {
19         "Monday" -> food = "flakes"
20         "Tuesday" -> food = "pellets"
21         "Wednesday" -> food = "redworms"
22         "Thursday" -> food = "granules"
23         "Friday" -> food = "mosquitoes"
24         "Saturday" -> food = "lettuce"
25         "Sunday" -> food = "plankton"
26     }
27     return food
28 }
```



Mô tả:

Tương tự như trên, nhưng 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 }

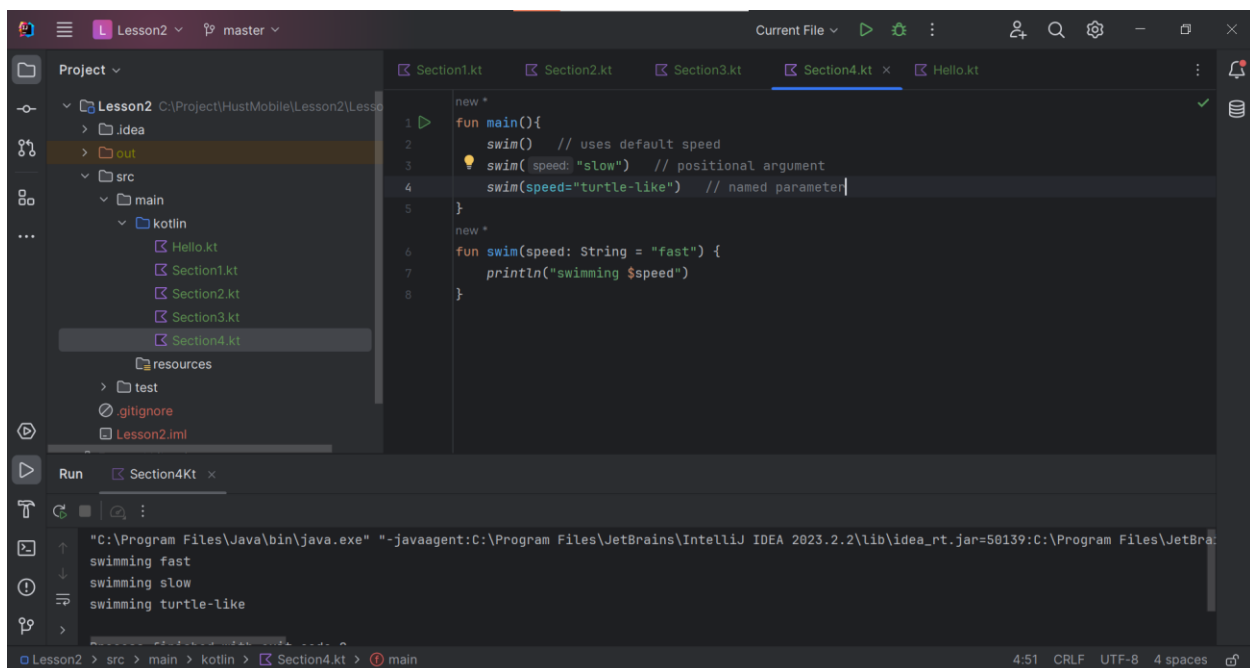
```

Mô tả:

Để code ngắn gọn hơn, ta có thể trực tiếp return về hàm kết quả của mệnh đề when. Ngoài ra, ta thêm trường hợp default là trả về nothing để tránh gặp phải các trường hợp ngoại lệ.

4. Explore default values and compact functions

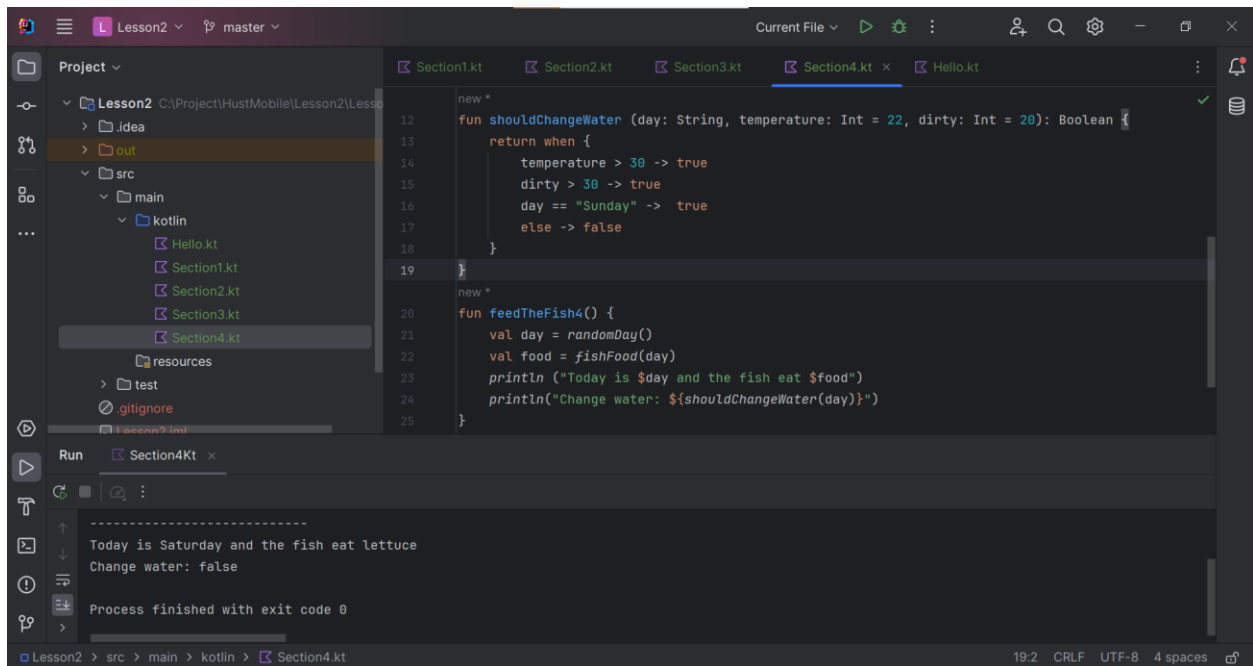
4.1. Create a default value for a parameter



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:

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

4.3. Make compact functions

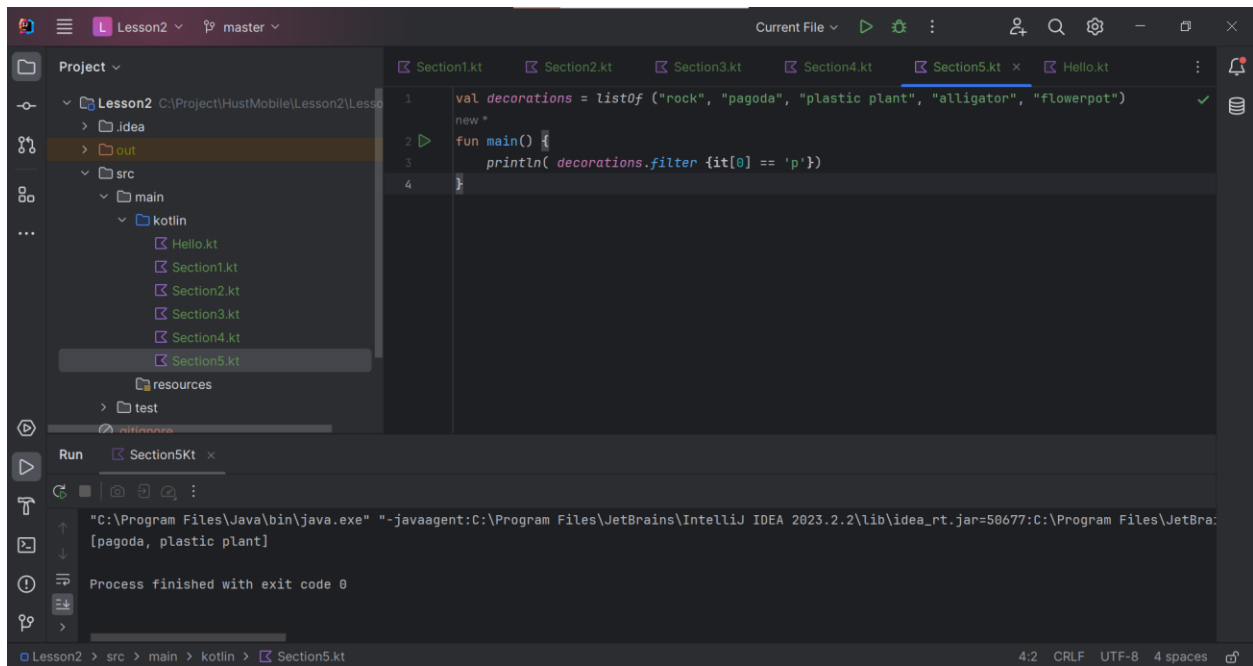


Mô tả:

Khi hàm chỉ là một phép tính và trả về giá trị phép tính đó, ta có thể sử dụng cách viết ngắn gọn hơn.

5. Get started with filters

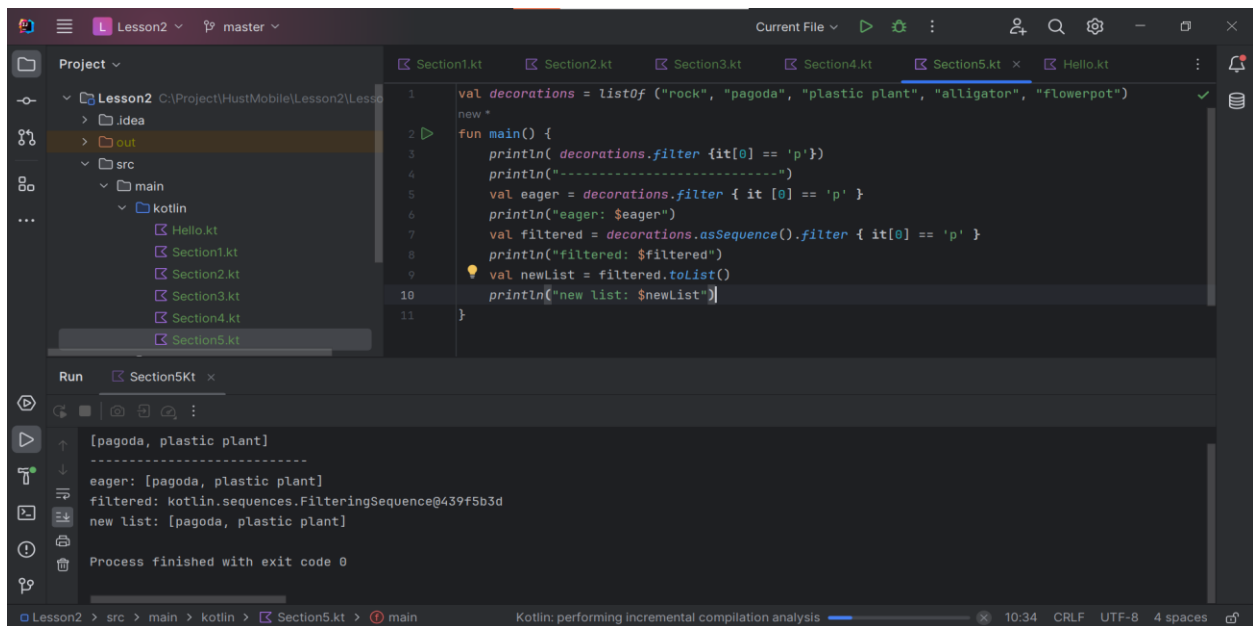
5.1. Create a filter



Mô tả:

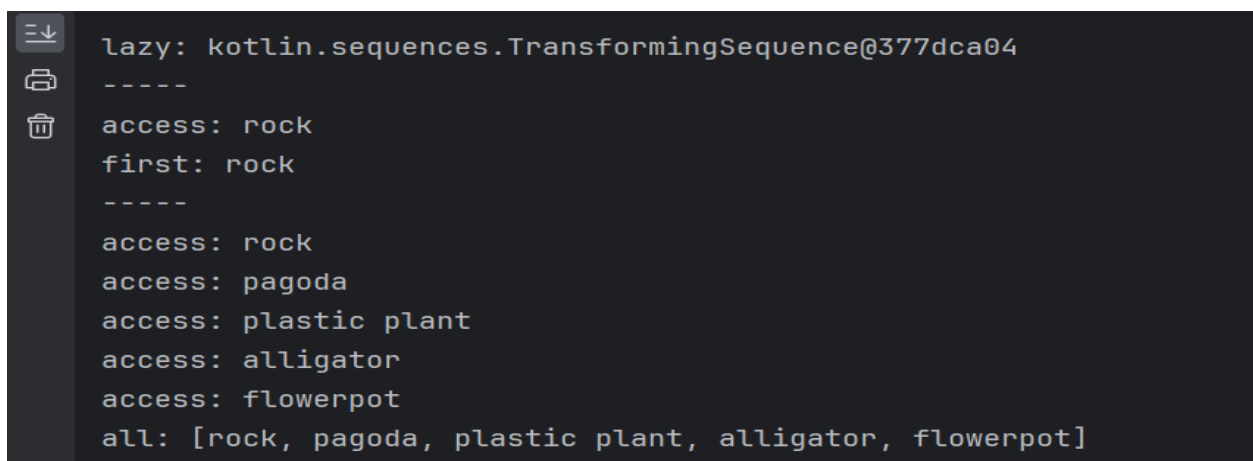
Phương thức filter cho phép kiểm tra các phần tử trong một danh sách, biến it là tham chiếu của mỗi phần tử, it[0] sẽ trả về giá trị đầu tiên của phần tử. Hàm trên sẽ trả về các phần tử bắt đầu bằng “p”.

5.2. Compare eager and lazy filters



Mô tả:

Eager sẽ load hết cùng lúc, Lazy thì sẽ load những gì cần thiết. Điều này có thể cải thiện hiệu năng hệ thống.



The screenshot shows the IntelliJ IDEA interface with a project named 'Lesson2'. The 'Project' view on the left shows the file structure: Lesson2 > src > main > kotlin > Section5.kt. The editor displays the following Kotlin code in Section5.kt:

```
println("-----")
val lazyMap2 = decorations.asSequence().filter {it[0] == 'p'}.map { it: String
    println("access: $it")
    it "map"
}
println("filtered: ${lazyMap2.toList()}")

println("-----")
val mysports = listOf("basketball", "fishing", "running")
val myplayers = listOf("LeBron James", "Ernest Hemingway", "Usain Bolt")
val mycities = listOf("Los Angeles", "Chicago", "Jamaica")
val myList = listOf(mysports, myplayers, mycities) // list of lists
println("Flat: ${myList.flatten()}")
```

The 'Run' view at the bottom shows the output of the program:

```
access: pagoda
access: plastic plant
filtered: [pagoda, plastic plant]
Flat: [basketball, fishing, running, LeBron James, Ernest Hemingway, Usain Bolt, Los Angeles, Chicago, Jamaica]
```

Process finished with exit code 0

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

The screenshot shows the IntelliJ IDEA interface with a project named 'Lesson2'. The 'Project' view on the left shows the file structure: Lesson2 > src > main > kotlin > Section6.kt. The editor displays the following Kotlin code in Section6.kt:

```
fun main() {
    var dirtyLevel = 20
    val waterFilter1 = { dirty : Int -> dirty / 2 }
    val waterFilter2: (Int) -> Int = { dirty -> dirty / 2 }
    println("waterFilter1: " + waterFilter1(dirtyLevel))
    println("waterFilter2: " + waterFilter2(dirtyLevel))
}
```

The 'Run' view at the bottom shows the output of the program:

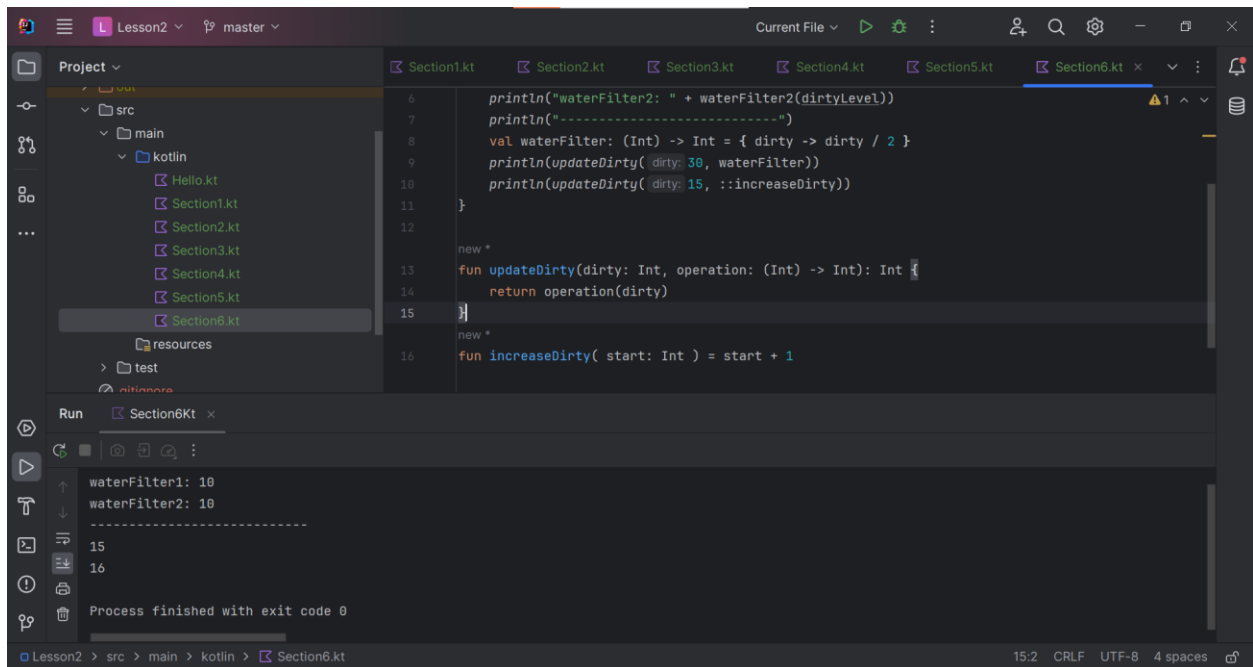
```
waterFilter1: 10
waterFilter2: 10
```

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



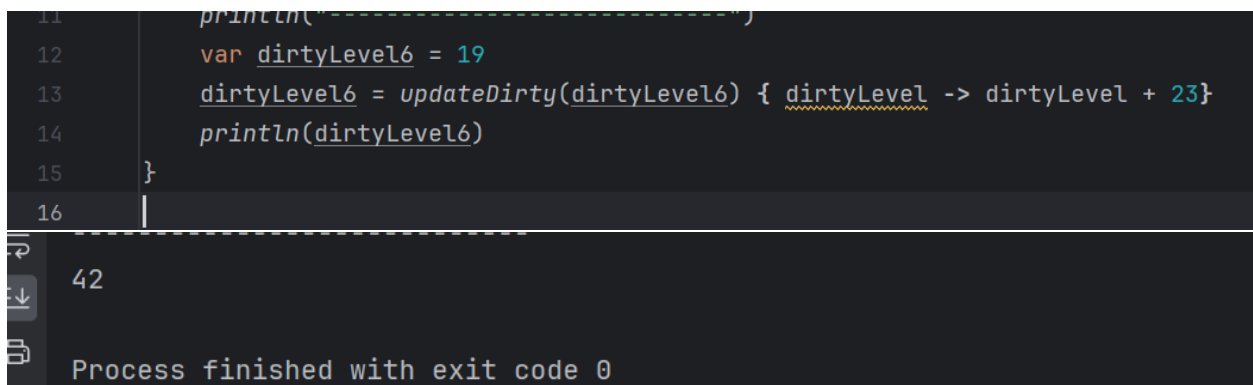
```
Project
├── src
│   ├── main
│   │   └── kotlin
│   │       ├── Hello.kt
│   │       ├── Section1.kt
│   │       ├── Section2.kt
│   │       ├── Section3.kt
│   │       ├── Section4.kt
│   │       ├── Section5.kt
│   │       └── Section6.kt
│   └── resources
└── test

Run: Section6Kt
waterFilter1: 10
waterFilter2: 10
-----
15
16
Process finished with exit code 0
```

Mô tả:

Một hàm bậc cao là một hàm lấy các hàm khác làm tham số hoặc là một hàm trả về một hàm khác. Bạn có thể truyền lambda cho một hàm bậc cao lấy một hàm làm đối số.

Hàm bạn truyền không nhất thiết phải là lambda; thay vào đó, nó có thể là một hàm được đặt tên thông thường. Để chỉ định đối số là một hàm thông thường, hãy sử dụng toán tử `::`. Theo cách này, Kotlin biết rằng bạn đang truyền tham chiếu hàm dưới dạng đối số, chứ không phải cố gắng gọi hàm.



```
11      println("-----")
12      var dirtyLevel6 = 19
13      dirtyLevel6 = updateDirty(dirtyLevel6) { dirtyLevel -> dirtyLevel + 23}
14      println(dirtyLevel6)
15  }
16

42
Process finished with exit code 0
```