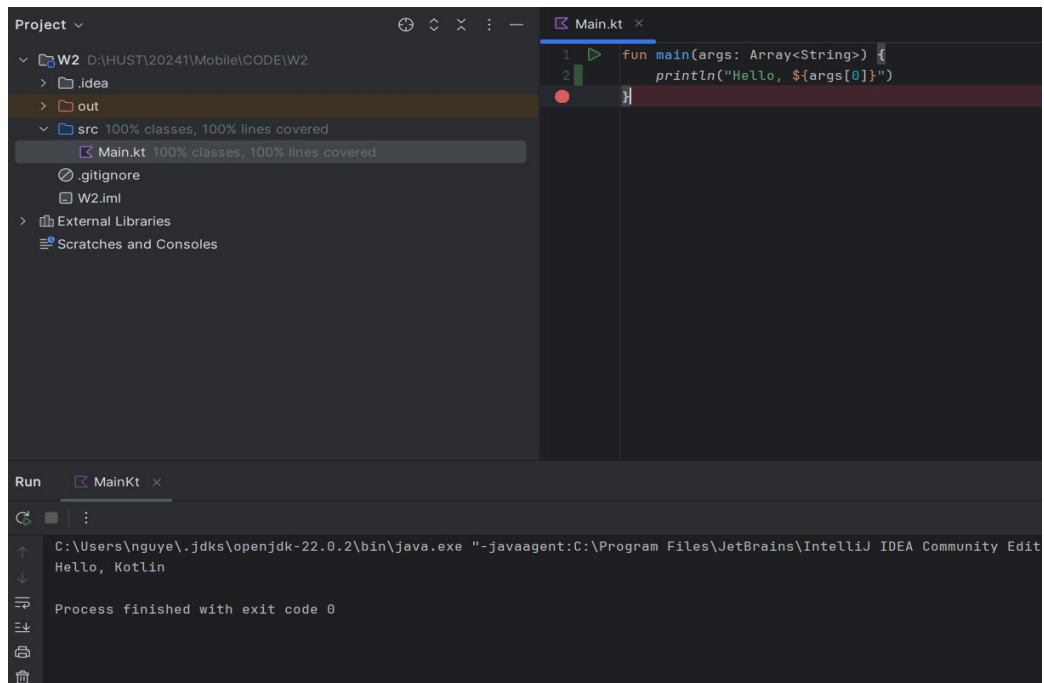
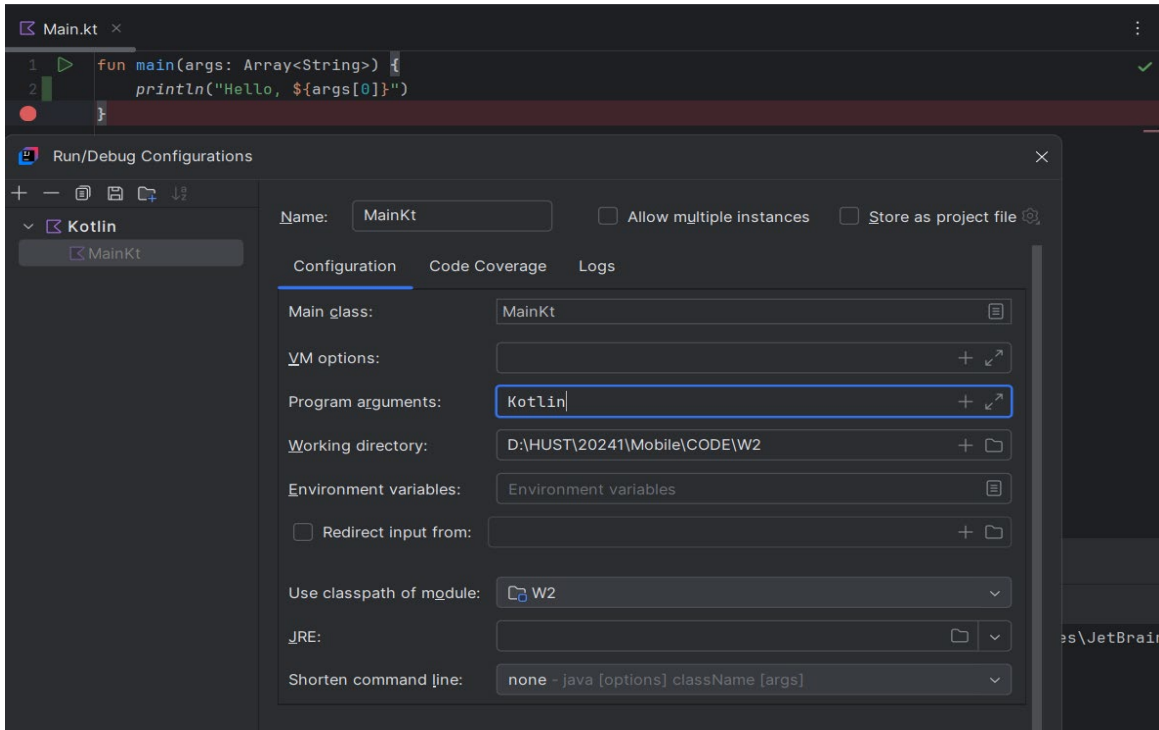


Name: Nguyễn Văn Cường – Student ID: 20215006

Report - Lesson 2: Functions

1. Explore the main() function



- Giải thích: Đoạn mã Kotlin này thực hiện các bước sau:

+ fun main(args: Array<String>) : Đây là hàm `main`, nơi chương trình bắt đầu. Nó nhận một mảng các chuỗi (`Array<String>`) từ dòng lệnh.

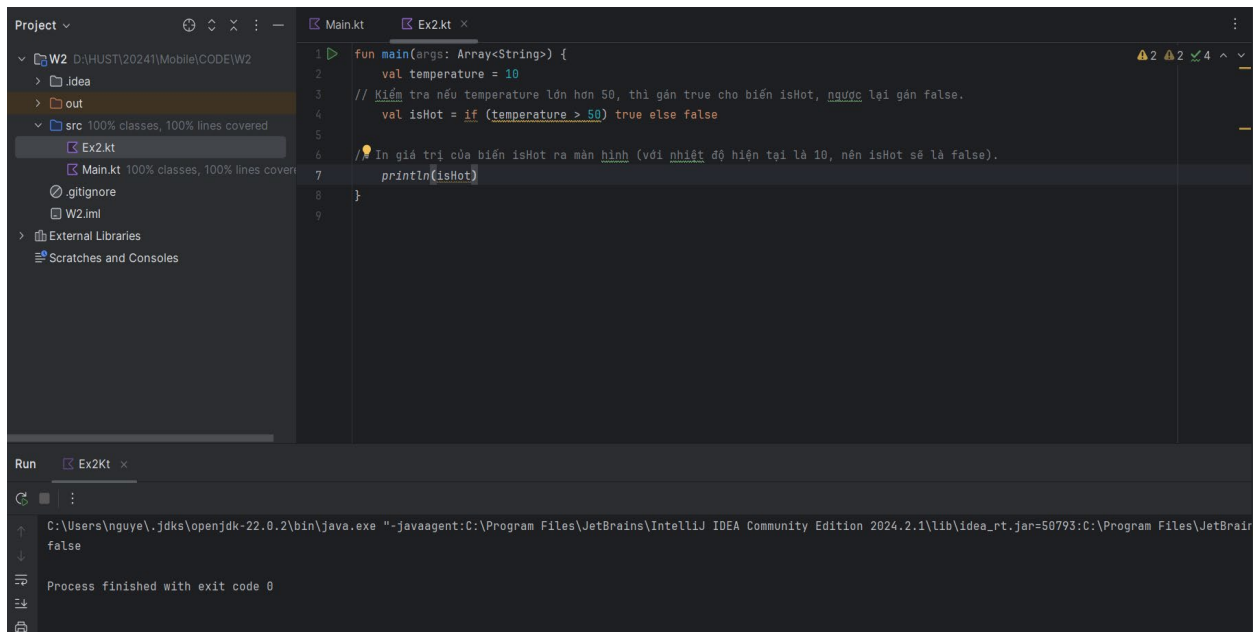
+ println("Hello, \${args[0]}") : In ra chuỗi "Hello, " kèm theo giá trị của phần tử đầu tiên trong mảng args (chính là args[0]). Dấu \${} được sử dụng để chèn giá trị của biểu thức vào chuỗi.

*****Từ bài số 2 mình sẽ chú thích vào code*****

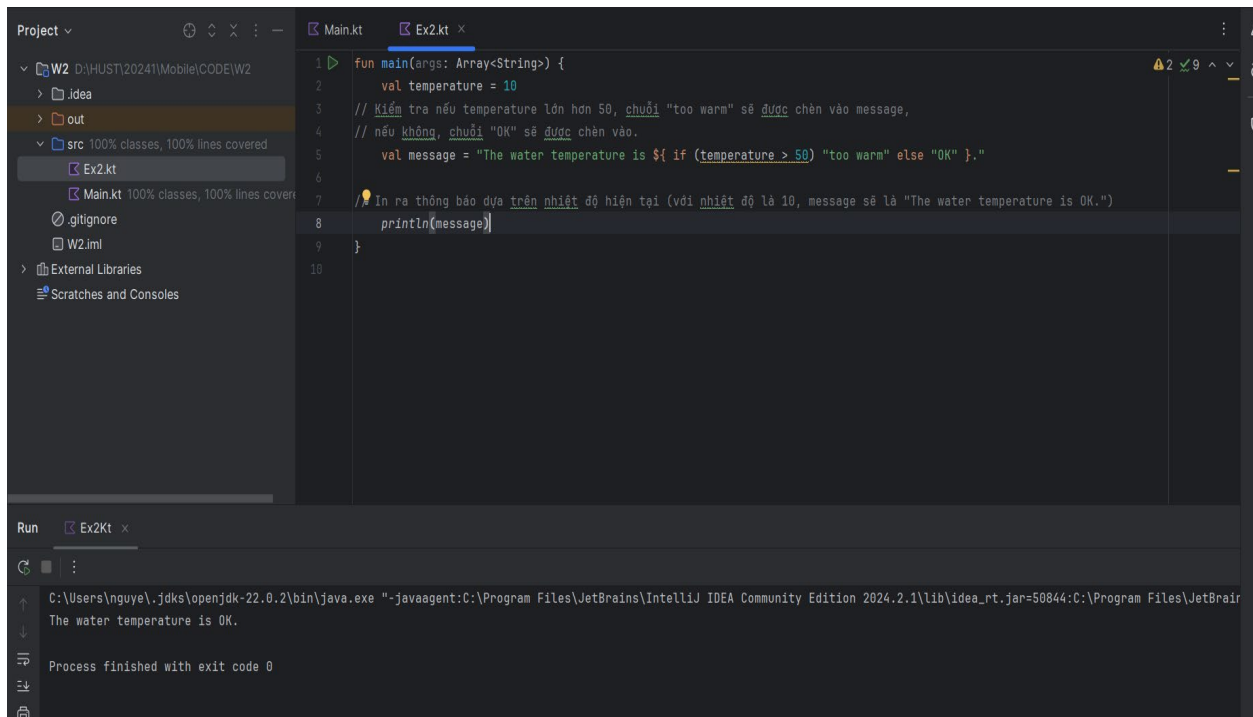
2. Learn why (almost) everything has a value

2.1.

2.2.



2.3.



3. Learn more about functions

3.1

```
Project ▾
  ▾ W2 D:\HUST\20241\Mobile\CODE\W2
    ▾ idea
      ▾ out
        ▾ src 100% classes, 100% lines covered
          ▾ Ex2.kt
          ▾ Ex3.kt
          ▾ Main.kt 100% classes, 100% lines covered
          ▾ .gitignore
          ▾ W2.iml
        ▾ External Libraries
        ▾ Scratches and Consoles

1 import java.util.* // Import thư viện để sử dụng lớp Random
2
3 // Hàm randomDay() trả về một ngày ngẫu nhiên trong tuần.
4 fun randomDay(): String {
5     // Mảng chứa các ngày trong tuần
6     val week = arrayOf("Monday", "Tuesday", "Wednesday", "Thursday",
7         "Friday", "Saturday", "Sunday")
8     // Sử dụng Random().nextInt() để chọn ngẫu nhiên một chỉ số của mảng và trả về ngày tương ứng.
9     return week[Random().nextInt(week.size)]
10 }
11
12 // Hàm feedTheFish() sẽ gọi hàm randomDay() để lấy một ngày ngẫu nhiên và in thông điệp về việc cho cá ăn.
13 fun feedTheFish() {
14     val day = randomDay() // Lấy một ngày ngẫu nhiên trong tuần
15     val food = "pellets" // Thức ăn mặc định là "pellets"
16     println("Today is $day and the fish eat $food") // In ra ngày và thức ăn cho cá
17 }
18
19 // Hàm main() là điểm khởi đầu của chương trình, gọi hàm feedTheFish().
20 fun main(args: Array<String>) {
21     feedTheFish() // Gọi hàm cho cá ăn
22 }
```

Run ▾ Ex3.kt ▾

C:\Users\nguyen\jdk\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.jar=51093:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\bin" -Didea.config.path=C:\Users\nguyen\AppData\Local\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\config -Didea.system.path=C:\Users\nguyen\AppData\Local\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\system -Didea.version=2024.2.1 Today is Wednesday and the fish eat pellets

Process finished with exit code 0

3.2

```
1 import java.util.* // Import thư viện để sử dụng lớp Random
2 // Hàm randomDay() trả về một ngày ngẫu nhiên trong tuần.
3 fun randomDay(): String {
4     // Mảng chứa các ngày trong tuần
5     val week = arrayOf("Monday", "Tuesday", "Wednesday", "Thursday",
6         "Friday", "Saturday", "Sunday")
7     // Sử dụng Random().nextInt() để chọn ngẫu nhiên một chỉ số của mảng và trả về ngày tương ứng.
8     return week[Random().nextInt(week.size)]
9 }
10
11 fun fishFood(day: String): String {
12     // Biến 'food' sẽ lưu trữ loại thức ăn cho cá tùy thuộc vào ngày.
13     var food = ""
14
15     // Sử dụng biểu thức 'when' để xác định loại thức ăn theo từng ngày trong tuần.
16     when (day) {
17         "Monday" -> food = "flakes"
18         "Tuesday" -> food = "pellets"
19         "Wednesday" -> food = "redworms"
20         "Thursday" -> food = "granules"
21         "Friday" -> food = "mosquitoes"
22         "Saturday" -> food = "lettuce"
23         "Sunday" -> food = "plankton"
24     }
25 }
```

```

25     // Trả về loại thức ăn tương ứng với ngày.
26     return food
27 }
28
29 ~ fun feedTheFish() {
30     // Lấy một ngày ngẫu nhiên bằng cách gọi hàm randomDay().
31     val day = randomDay()
32     // Dựa trên ngày ngẫu nhiên, lấy loại thức ăn từ hàm fishFood().
33     val food = fishFood(day)
34     // In ra thông báo về ngày hôm nay và loại thức ăn mà cá sẽ ăn.
35     println("Today is $day and the fish eat $food")
36 }
37
38 // Hàm main() là điểm khởi đầu của chương trình, gọi hàm feedTheFish().
39 ~ fun main(args: Array<String>) {
40     feedTheFish() // Gọi hàm cho cá ăn
41 }
42

```

Kết quả:

```

C:\Users\nguye\.jdk\openjdk-22.0.2\bin\java.exe "-javaagent:C:\P
Today is Tuesday and the fish eat pellets

Process finished with exit code 0

```

3.3

Do chỉ thay đổi hàm fishFood nên mình sẽ chỉ chụp đoạn code kèm giải thích của riêng hàm đó, mấy hàm còn lại tương tự

```

fun fishFood(day: String): String {
    // Khai báo biến food để lưu loại thức ăn.
    val food: String

    // Sử dụng biểu thức 'when' để xác định loại thức ăn tùy vào ngày.
    when (day) {
        "Monday" -> food = "flakes"           // Thứ Hai: flakes
        "Wednesday" -> food = "redworms"     // Thứ Tư: redworms
        "Thursday" -> food = "granules"      // Thứ Năm: granules
        "Friday" -> food = "mosquitoes"     // Thứ Sáu: mosquitoes
        "Sunday" -> food = "plankton"        // Chủ Nhật: plankton
        else -> food = "nothing"             // Những ngày còn lại: không có thức ăn
    }

    // Trả về loại thức ăn được xác định.
    return food
}

```

Kết quả:

```
C:\Users\nguye\.jdk\openjdk-22.0.2\bin\java.exe "-javaagent:C:\P
Today is Tuesday and the fish eat nothing

Process finished with exit code 0
```

3.4

Cũng chỉ thay đổi hàm fishFood nên mình làm như 3.3

```
10 fun fishFood(day: String): String {
11     // Sử dụng 'when' để trả về loại thức ăn cho cá dựa trên ngày.
12     return when (day) {
13         "Monday" -> "flakes"           // Nếu ngày là Thứ Hai, trả về "flakes".
14         "Wednesday" -> "redworms"      // Nếu ngày là Thứ Tư, trả về "redworms".
15         "Thursday" -> "granules"       // Nếu ngày là Thứ Năm, trả về "granules".
16         "Friday" -> "mosquitoes"      // Nếu ngày là Thứ Sáu, trả về "mosquitoes".
17         "Sunday" -> "plankton"         // Nếu ngày là Chủ Nhật, trả về "plankton".
18         else -> "nothing"             // Nếu ngày không phải là các ngày trên, trả về "nothing".
19     }
20 }
21
```

⇒ Tổng quan lại chương trình và kết quả chạy:

```
1 import java.util.* // Import thư viện để sử dụng lớp Random
2 // Hàm randomDay() trả về một ngày ngẫu nhiên trong tuần.
3 fun randomDay(): String {
4     // Mảng chứa các ngày trong tuần
5     val week = arrayOf("Monday", "Tuesday", "Wednesday", "Thursday",
6         "Friday", "Saturday", "Sunday")
7     // Sử dụng Random().nextInt() để chọn ngẫu nhiên một chỉ số của mảng và trả về ngày tương ứng.
8     return week[Random().nextInt(week.size)]
9 }
10
11 fun fishFood(day: String): String {
12     // Sử dụng 'when' để trả về loại thức ăn cho cá dựa trên ngày.
13     return when (day) {
14         "Monday" -> "flakes"           // Nếu ngày là Thứ Hai, trả về "flakes".
15         "Wednesday" -> "redworms"      // Nếu ngày là Thứ Tư, trả về "redworms".
16         "Thursday" -> "granules"       // Nếu ngày là Thứ Năm, trả về "granules".
17         "Friday" -> "mosquitoes"      // Nếu ngày là Thứ Sáu, trả về "mosquitoes".
18         "Sunday" -> "plankton"         // Nếu ngày là Chủ Nhật, trả về "plankton".
19         else -> "nothing"             // Nếu ngày không phải là các ngày trên, trả về "nothing".
20     }
21 }
```

```

22
23 fun feedTheFish() {
24     // Lấy một ngày ngẫu nhiên bằng cách gọi hàm randomDay().
25     val day = randomDay()
26     // Dựa trên ngày ngẫu nhiên, lấy loại thức ăn từ hàm fishFood().
27     val food = fishFood(day)
28     // In ra thông báo về ngày hôm nay và loại thức ăn mà cá sẽ ăn.
29     println("Today is $day and the fish eat $food")
30 }
31
32 // Hàm main() là điểm khởi đầu của chương trình, gọi hàm feedTheFish().
33 fun main(args: Array<String>) {
34     feedTheFish() // Gọi hàm cho cá ăn
35 }
36

```

Kết quả:

```

C:\Users\nguye\.jdk\openjdk-22.0.2\bin\java.exe "
Today is Wednesday and the fish eat redworms

Process finished with exit code 0

```

4. Explore default values and compact functions

4.1

The screenshot shows the IntelliJ IDEA IDE with a project named 'W2'. The file 'Ex4.kt' is open, showing the following code:

```

1 fun swim(speed: String = "fast") {
2     // Hàm swim nhận tham số 'speed' với giá trị mặc định là "fast".
3     println("swimming $speed") // In ra thông báo "swimming" kèm theo tốc độ.
4 }
5
6 fun main() {
7     swim() // Gọi hàm swim mà không truyền tham số, sử dụng giá trị mặc định "fast".
8     swim(speed = "slow") // Gọi hàm swim với tham số "slow", đây là đối số vị trí.
9     swim(speed = "turtle-like") // Gọi hàm swim với tham số có tên, chỉ định tốc độ là "turtle-like".
10 }
11

```

The Run console at the bottom shows the output of the program:

```

C:\Users\nguye\.jdk\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.jar=51517:C:\Program File
swimming fast
swimming slow
swimming turtle-like
Process finished with exit code 0

```

4.2

Do các hàm kia giống Ex3 nên mình sẽ thêm hàm shouldChangeWater vào đầu Ex3 và chạy code


```

1 import java.util.* // Import thư viện để sử dụng lớp Random
2 fun shouldChangeWater(day: String, temperature: Int = 22, dirty: Int = 20): Boolean {
3     return when {
4         temperature > 30 -> true // Nếu nhiệt độ lớn hơn 30, trả về true (cần thay nước).
5         dirty > 30 -> true // Nếu độ bẩn lớn hơn 30, trả về true (cần thay nước).
6         day == "Sunday" -> true // Nếu hôm nay là Chủ Nhật, trả về true (cần thay nước).
7         else -> false // Trong các trường hợp khác, trả về false (không cần thay nước).
8     }
9 }
10 // Hàm randomDay() trả về một ngày ngẫu nhiên trong tuần.
11 fun randomDay(): String {
12     // Mảng chứa các ngày trong tuần
13     val week = arrayOf("Monday", "Tuesday", "Wednesday", "Thursday",
14         "Friday", "Saturday", "Sunday")
15     // Sử dụng Random().nextInt() để chọn ngẫu nhiên một chỉ số của mảng và trả về ngày tương ứng.
16     return week[Random().nextInt(week.size)]
17 }
18
19 fun fishFood(day: String): String {
20     // Sử dụng 'when' để trả về loại thức ăn cho cá dựa trên ngày.

```

Kết quả:

```
C:\Users\nguye\.jdk\openjdk-22.0.2\bin\java.exe "-java
Today is Wednesday and the fish eat redworms
Change water: false

Process finished with exit code 0
```

4.3

The screenshot displays the IntelliJ IDEA IDE interface. The top toolbar includes icons for Project, Main.kt, Ex2.kt, and Ex3.kt. The left sidebar shows the Project view with the following structure:

- W2 (D:\HUST\20241\Mobile\CODE\W2)
 - .idea
 - out
 - src
 - Ex2.kt
 - Ex3.kt (selected)
 - Main.kt
 - .gitignore
 - W2.iml
- External Libraries
- Scratches and Consoles

The main editor window shows the code in Ex3.kt:

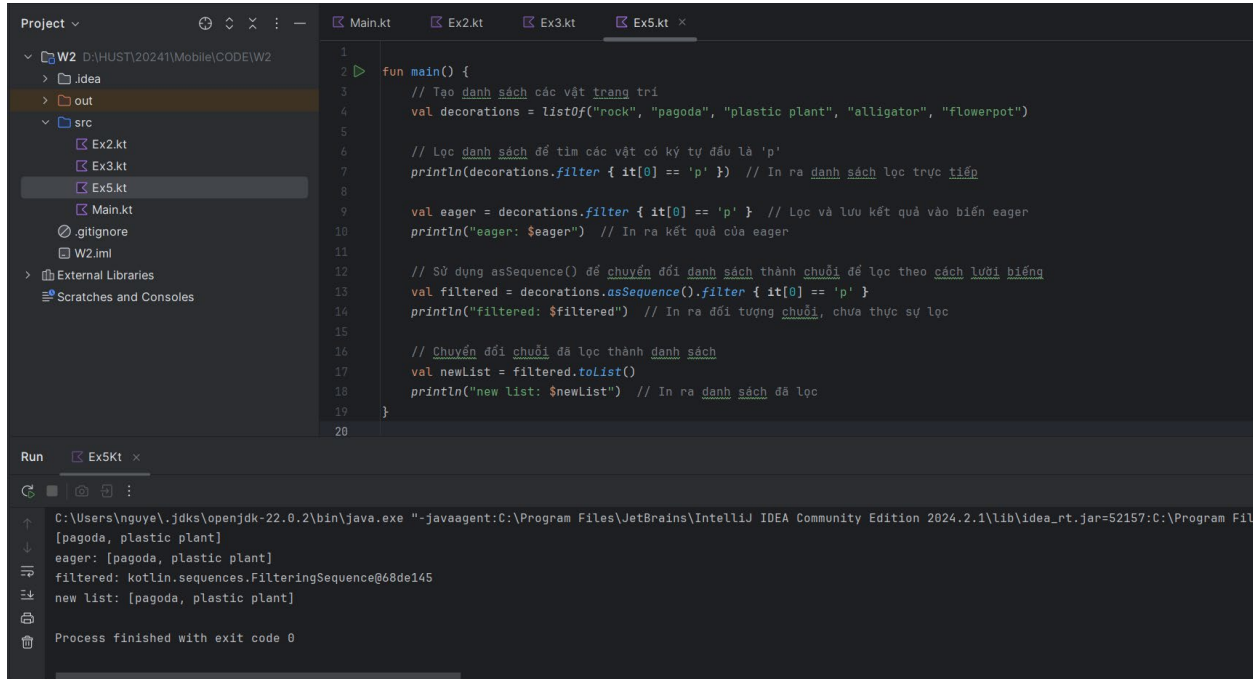
```
1 import java.util.* // Import thư viện để sử dụng lớp Random
2 fun isTooHot(temperature: Int) = temperature > 30 // Kiểm tra xem nhiệt độ có lớn hơn 30 không.
3
4 fun isDirty(dirty: Int) = dirty > 30 // Kiểm tra xem độ bẩn có lớn hơn 30 không.
5
6 fun isSunday(day: String) = day == "Sunday" // Kiểm tra xem ngày có phải là Chủ Nhật không.
7
8 fun shouldChangeWater(day: String, temperature: Int = 22, dirty: Int = 20): Boolean {
9     return when {
10         isTooHot(temperature) -> true // Nếu nhiệt độ quá nóng, cần thay nước.
11         isDirty(dirty) -> true // Nếu nước bẩn, cần thay nước.
12         isSunday(day) -> true // Nếu hôm nay là Chủ Nhật, cần thay nước.
13         else -> false // Không cần thay nước trong các trường hợp khác.
14     }
15 }
16
17 // Hàm randomDay() trả về một ngày ngẫu nhiên trong tuần.
18 fun randomDay(): String {
19     // Mảng chứa các ngày trong tuần
20     val week = arrayOf("Monday", "Tuesday", "Wednesday", "Thursday",
```

The Run tab at the bottom shows the execution output:

```
C:\Users\nguye\.jdk\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.jar=51934:C:\Program Fil
Today is Sunday and the fish eat plankton
Change water: true

Process finished with exit code 0
```


5. Get started with filters



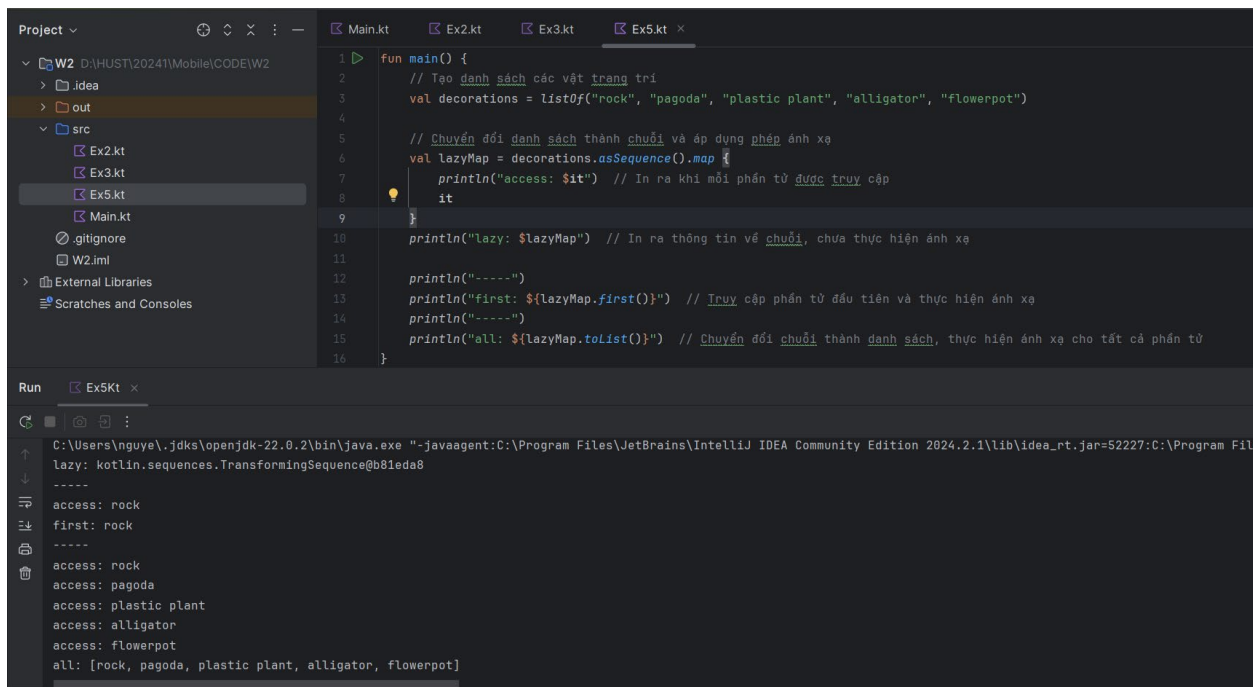
The screenshot shows the IntelliJ IDEA IDE with a Kotlin project. The left sidebar displays the project structure with files like Main.kt, Ex2.kt, Ex3.kt, and Ex5.kt. The main editor shows the code for Ex5.kt, which demonstrates the use of `filter` and `asSequence` functions. The Run console at the bottom shows the output of the program.

```
1 fun main() {
2     // Tạo danh sách các vật trang trí
3     val decorations = listOf("rock", "pagoda", "plastic plant", "alligator", "flowerpot")
4
5     // Lọc danh sách để tìm các vật có ký tự đầu là 'p'
6     println(decorations.filter { it[0] == 'p' }) // In ra danh sách lọc trực tiếp
7
8     val eager = decorations.filter { it[0] == 'p' } // Lọc và lưu kết quả vào biến eager
9     println("eager: $eager") // In ra kết quả của eager
10
11     // Sử dụng asSequence() để chuyển đổi danh sách thành chuỗi để lọc theo cách lười biếng
12     val filtered = decorations.asSequence().filter { it[0] == 'p' }
13     println("filtered: $filtered") // In ra đối tượng chuỗi, chưa thực sự lọc
14
15     // Chuyển đổi chuỗi đã lọc thành danh sách
16     val newList = filtered.toList()
17     println("new list: $newList") // In ra danh sách đã lọc
18 }
19
20
```

Run console output:

```
C:\Users\nguye\.jdk\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.jar=52157:C:\Program Fil
[pagoda, plastic plant]
eager: [pagoda, plastic plant]
filtered: kotlin.sequences.FilteringSequence@68de145
new list: [pagoda, plastic plant]
Process finished with exit code 0
```

Sử dụng lazyMap:



The screenshot shows the IntelliJ IDEA IDE with a Kotlin project. The left sidebar displays the project structure with files like Main.kt, Ex2.kt, Ex3.kt, and Ex5.kt. The main editor shows the code for Ex5.kt, which demonstrates the use of the `lazyMap` function. The Run console at the bottom shows the output of the program.

```
1 fun main() {
2     // Tạo danh sách các vật trang trí
3     val decorations = listOf("rock", "pagoda", "plastic plant", "alligator", "flowerpot")
4
5     // Chuyển đổi danh sách thành chuỗi và áp dụng phép ánh xạ
6     val lazyMap = decorations.asSequence().map {
7         println("access: $it") // In ra khi mỗi phần tử được truy cập
8         it
9     }
10     println("lazy: $lazyMap") // In ra thông tin về chuỗi, chưa thực hiện ánh xạ
11
12     println("-----")
13     println("first: ${lazyMap.first()}") // Truy cập phần tử đầu tiên và thực hiện ánh xạ
14     println("-----")
15     println("all: ${lazyMap.toList()}") // Chuyển đổi chuỗi thành danh sách, thực hiện ánh xạ cho tất cả phần tử
16 }
```

Run console output:

```
C:\Users\nguye\.jdk\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.jar=52227:C:\Program Fil
lazy: kotlin.sequences.TransformingSequence@b81eda8
-----
access: rock
-----
first: rock
-----
access: rock
access: pagoda
access: plastic plant
access: alligator
access: flowerpot
all: [rock, pagoda, plastic plant, alligator, flowerpot]
```

Sử dụng thêm với filter

The screenshot shows the IntelliJ IDEA interface with a Kotlin project. The left sidebar displays the project structure with files like `Ex2.kt`, `Ex3.kt`, `Ex5.kt`, and `Main.kt`. The main editor shows the code in `Ex5.kt`:

```
1 fun main() {
2     // Tạo danh sách các vật trang trí
3     val decorations = listOf("rock", "pagoda", "plastic plant", "alligator", "flowerpot")
4
5     // Chuyển đổi danh sách thành chuỗi chuỗi lười biếng, lọc và ánh xạ
6     val lazyMap2 = decorations.asSequence()
7         .filter { it[0] == 'p' } // Lọc các vật có ký tự đầu là 'p'
8         .map {
9             println("access: $it") // In ra khi mỗi phần tử được truy cập
10             it
11         }
12
13     println("-----")
14     println("filtered: ${lazyMap2.toList()}") // Chuyển đổi chuỗi thành danh sách, thực hiện lọc và ánh xạ
15 }
16
```

The Run console at the bottom shows the execution output:

```
C:\Users\nguye\.jids\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.jar=52306:C:\Program Fil
-----
access: pagoda
access: plastic plant
filtered: [pagoda, plastic plant]
Process finished with exit code 0
```

6. Get started with lambdas and higher-order functions

6.1

The screenshot shows the IntelliJ IDEA interface with a Kotlin project. The left sidebar displays the project structure with files like `Ex2.kt`, `Ex3.kt`, `Ex5.kt`, and `Ex6.kt`. The main editor shows the code in `Ex6.kt`:

```
1 fun main() {
2     var dirtyLevel = 20 // Khởi tạo biến mức độ bẩn của nước
3
4     val waterFilter = { dirty: Int -> Int } { dirty / 2 } // Định nghĩa hàm lambda để lọc nước
5     println(waterFilter(dirtyLevel)) // Gọi hàm lambda với mức độ bẩn và in kết quả
6 }

```

The Run console at the bottom shows the execution output:

```
C:\Users\nguye\.jids\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.jar=52404:C:\Program Fil
10
Process finished with exit code 0
```

6.2

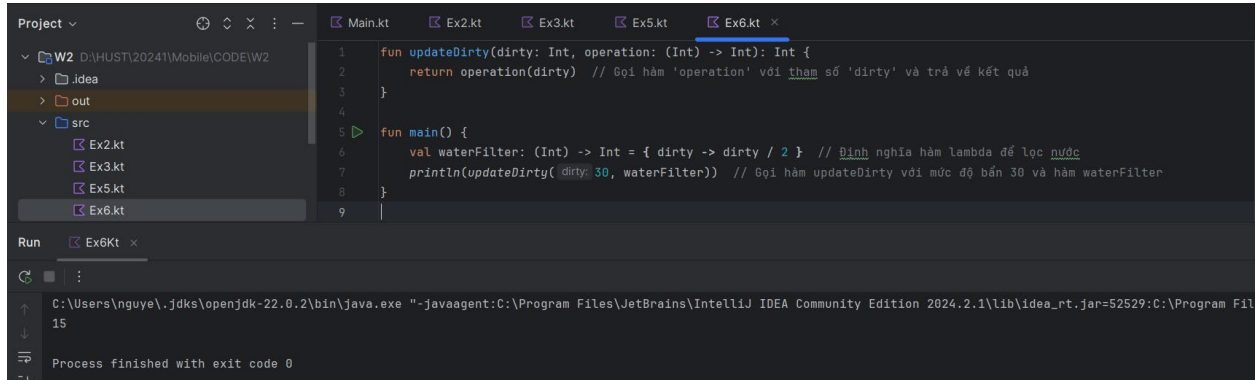
The screenshot shows the IntelliJ IDEA interface with a Kotlin project. The left sidebar displays the project structure with files like `Ex2.kt`, `Ex3.kt`, `Ex5.kt`, and `Ex6.kt`. The main editor shows the code in `Ex6.kt`:

```
1 fun main() {
2     var dirtyLevel = 20 // Khởi tạo biến mức độ bẩn của nước với giá trị 20
3
4     // Định nghĩa hàm lambda để lọc nước
5     // waterFilter là một biến kiểu hàm nhận vào một Int và trả về một Int
6     val waterFilter: (Int) -> Int = { dirty -> dirty / 2 }
7
8     // Gọi hàm lambda với mức độ bẩn và in kết quả
9     println(waterFilter(dirtyLevel)) // In ra giá trị trả về của hàm waterFilter
10 }
11
```

The Run console at the bottom shows the execution output:

```
C:\Users\nguye\.jids\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.jar=52427:C:\Program Fil
10
Process finished with exit code 0
```

6.3



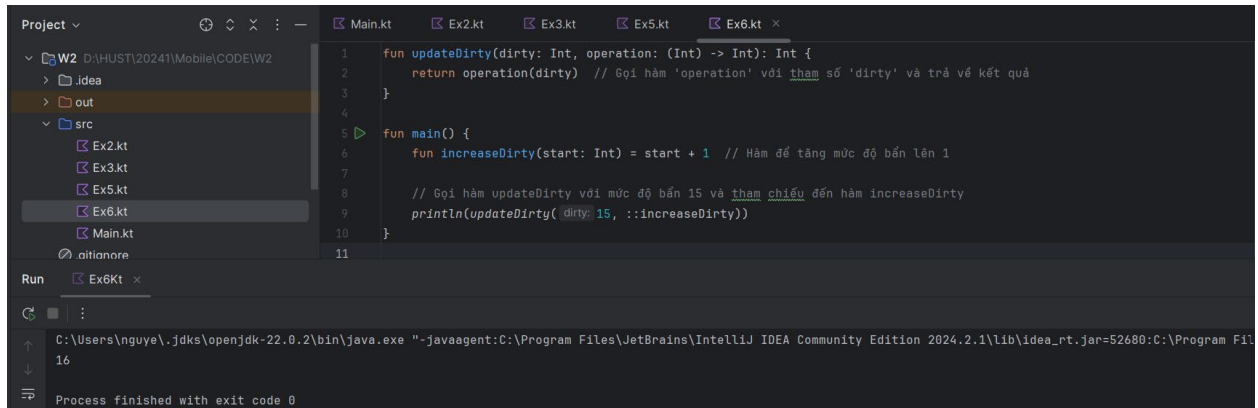
```
Project ▾
  ▾ W2 D:\HUST\20241\Mobile\CODE\W2
    ▾ .idea
    ▾ out
    ▾ src
      Ex2.kt
      Ex3.kt
      Ex5.kt
      Ex6.kt

Run Ex6.kt ×

C:\Users\nguye\jdk\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.jar=52529:C:\Program Fil
15
Process finished with exit code 0
```

```
1 fun updateDirty(dirty: Int, operation: (Int) -> Int): Int {
2     return operation(dirty) // Gọi hàm 'operation' với tham số 'dirty' và trả về kết quả
3 }
4
5 fun main() {
6     val waterFilter: (Int) -> Int = { dirty -> dirty / 2 } // Định nghĩa hàm lambda để lọc nước
7     println(updateDirty(dirty: 30, waterFilter)) // Gọi hàm updateDirty với mức độ bẩn 30 và hàm waterFilter
8 }
9
```

6.4



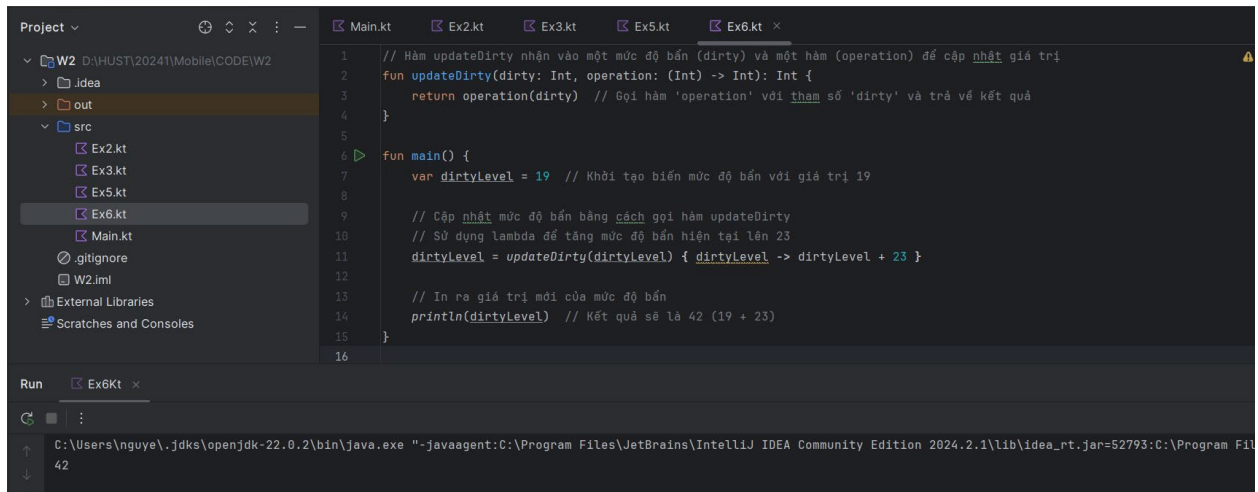
```
Project ▾
  ▾ W2 D:\HUST\20241\Mobile\CODE\W2
    ▾ .idea
    ▾ out
    ▾ src
      Ex2.kt
      Ex3.kt
      Ex5.kt
      Ex6.kt
      Main.kt
    .gitignore
    W2.iml
  ▾ External Libraries
  ▾ Scratches and Consoles

Run Ex6.kt ×

C:\Users\nguye\jdk\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.jar=52680:C:\Program Fil
16
Process finished with exit code 0
```

```
1 fun updateDirty(dirty: Int, operation: (Int) -> Int): Int {
2     return operation(dirty) // Gọi hàm 'operation' với tham số 'dirty' và trả về kết quả
3 }
4
5 fun main() {
6     fun increaseDirty(start: Int) = start + 1 // Hàm để tăng mức độ bẩn lên 1
7
8     // Gọi hàm updateDirty với mức độ bẩn 15 và tham chiếu đến hàm increaseDirty
9     println(updateDirty(dirty: 15, ::increaseDirty))
10 }
11
```

6.5



```
Project ▾
  ▾ W2 D:\HUST\20241\Mobile\CODE\W2
    ▾ .idea
    ▾ out
    ▾ src
      Ex2.kt
      Ex3.kt
      Ex5.kt
      Ex6.kt
      Main.kt
    .gitignore
    W2.iml
  ▾ External Libraries
  ▾ Scratches and Consoles

Run Ex6.kt ×

C:\Users\nguye\jdk\openjdk-22.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.2.1\lib\idea_rt.jar=52793:C:\Program Fil
42
Process finished with exit code 0
```

```
1 // Hàm updateDirty nhận vào một mức độ bẩn (dirty) và một hàm (operation) để cập nhật giá trị
2 fun updateDirty(dirty: Int, operation: (Int) -> Int): Int {
3     return operation(dirty) // Gọi hàm 'operation' với tham số 'dirty' và trả về kết quả
4 }
5
6 fun main() {
7     var dirtyLevel = 19 // Khởi tạo biến mức độ bẩn với giá trị 19
8
9     // Cập nhật mức độ bẩn bằng cách gọi hàm updateDirty
10    // Sử dụng lambda để tăng mức độ bẩn hiện tại lên 23
11    dirtyLevel = updateDirty(dirtyLevel) { dirtyLevel -> dirtyLevel + 23 }
12
13    // In ra giá trị mới của mức độ bẩn
14    println(dirtyLevel) // Kết quả sẽ là 42 (19 + 23)
15 }
16
```