**Chu Anh Lợi – 20215280**

*Lesson 2*

[1. Explore the main() function](https://developer.android.com/codelabs/android-development-kotlin-2.1#0)

Step 1: Create a Kotlin file

Tạo file Kotlin Hello.kt

A screenshot of a computer program

Description automatically generated

Step 2: Add code and run your program

Chạy thử codeA screenshot of a computer

Description automatically generated

Step 3: Pass arguments to main()

Truyền đối số vào hàm mainA screenshot of a computer

Description automatically generated

Step 4: Change the code to use a string template

Tempalte chuỗi để codeA screenshot of a computer

Description automatically generated

[2. Learn why (almost) everything has a value](https://developer.android.com/codelabs/android-development-kotlin-2.1#1)  
Một số ngôn ngữ lập trình chỉ có các câu lệnh, là những dòng mã không có giá trị. Trong Kotlin, hầu hết mọi thứ đều là biểu thức và có một giá trị — ngay cả khi giá trị đó là kotlin.Unit.A screenshot of a computer program

Description automatically generated

[3. Learn more about functions](https://developer.android.com/codelabs/android-development-kotlin-2.1#2)

Step 1: Create some functions

Viết một hàm gọi là feedTheFish() để gọi hàm randomDay() nhằm lấy một ngày ngẫu nhiên trong tuần. Sử dụng template chuỗi để in ra loại thức ăn mà cá sẽ ăn vào ngày đó. Hiện tại, cá ăn cùng một loại thức ăn mỗi ngày. Thêm một hàm randomDay() trong tệp Hello.kt để chọn một ngày ngẫu nhiên từ một mảng và trả về giá trị đó. Hàm nextInt() nhận một giới hạn số nguyên, giới hạn số từ Random() từ 0 đến 6 để tương ứng với mảng tuần. Các hàm Random() và nextInt() được định nghĩa trong java.util.\*. Ở đầu tệp, thêm phần import cần thiết.A screenshot of a computer

Description automatically generated

Step 2: Use a when expression

Mở rộng thêm, thay đổi mã để chọn thức ăn khác nhau cho từng ngày sử dụng biểu thức when. Câu lệnh when tương tự như switch trong các ngôn ngữ lập trình khác, nhưng when tự động ngắt ở cuối mỗi nhánh. Nó cũng đảm bảo mã của bạn bao quát tất cả các nhánh nếu bạn đang kiểm tra một enum.

Trong tệp Hello.kt, thêm một hàm gọi là fishFood() nhận một tham số là day kiểu String và trả về thức ăn của cá cho ngày đó dưới dạng String. Sử dụng when(), để mỗi ngày cá nhận được một loại thức ăn cụ thể. Chạy chương trình của bạn vài lần để xem các kết quả khác nhau.

Thêm một nhánh mặc định cho biểu thức when bằng cách sử dụng else. Để kiểm tra, để đảm bảo nhánh mặc định được chọn đôi khi trong chương trình của bạn, hãy loại bỏ các nhánh Tuesday và Saturday.

Có một nhánh mặc định đảm bảo rằng thức ăn sẽ có một giá trị trước khi được trả về, vì vậy nó không cần được khởi tạo nữa. Vì mã hiện chỉ gán một chuỗi cho food một lần, bạn có thể khai báo food với val thay vì var.

Vì mọi biểu thức đều có giá trị, bạn có thể làm mã này ngắn gọn hơn. Trả về giá trị của biểu thức when trực tiếp và loại bỏ biến food. Giá trị của biểu thức when là giá trị của biểu thức cuối cùng trong nhánh thỏa mãn điều kiện.

Phiên bản cuối cùng của chương trình của bạn sẽ trông giống như mã dưới đây.

A screenshot of a computer program

Description automatically generated

[4. Explore default values and compact functions](https://developer.android.com/codelabs/android-development-kotlin-2.1#3)

Step 1: Create a default value for a parameter

Trong Kotlin, bạn có thể truyền đối số theo tên của tham số. Bạn cũng có thể chỉ định các giá trị mặc định cho các tham số: nếu một đối số không được cung cấp bởi người gọi, giá trị mặc định sẽ được sử dụng. Sau này, khi bạn viết các phương thức (hàm thành viên), điều này có nghĩa là bạn có thể tránh việc phải viết nhiều phiên bản nạp chồng (overload) của cùng một phương thức.

Trong tệp Hello.kt, viết một hàm swim() với một tham số kiểu String tên là speed để in ra tốc độ bơi của cá. Tham số speed có giá trị mặc định là "fast".

Từ hàm main(), gọi hàm swim() theo ba cách. Đầu tiên gọi hàm sử dụng giá trị mặc định. Sau đó, gọi hàm và truyền tham số speed mà không có tên, rồi gọi hàm bằng cách đặt tên cho tham số speed.

A screenshot of a computer program

Description automatically generated

Step 2: Add required parameters

Nếu không chỉ định giá trị mặc định cho một tham số, thì đối số tương ứng phải luôn được truyền vào.

Trong tệp Hello.kt, viết một hàm shouldChangeWater() nhận ba tham số: day, temperature, và dirtyLevel. Hàm này trả về true nếu nước cần được thay, điều này xảy ra nếu là Chủ nhật, nếu nhiệt độ quá cao, hoặc nếu nước quá bẩn. Ngày trong tuần là tham số bắt buộc, nhưng nhiệt độ mặc định là 22, và mức độ bẩn mặc định là 20.

Sử dụng một biểu thức when không có đối số, điều này trong Kotlin hoạt động như một chuỗi các kiểm tra if/else if.

Gọi hàm shouldChangeWater() từ feedTheFish() và cung cấp ngày. Tham số day không có giá trị mặc định, vì vậy bạn phải chỉ định một đối số. Hai tham số còn lại của shouldChangeWater() có giá trị mặc định, nên bạn không cần phải truyền đối số cho chúng.

A screenshot of a computer program

Description automatically generated

Step 3: Make compact functions

Biểu thức `when` mà bạn viết ở bước trước gói gọn nhiều logic vào một lượng mã nhỏ. Nếu bạn muốn tách nó ra một chút, hoặc nếu các điều kiện cần kiểm tra phức tạp hơn, bạn có thể sử dụng một số biến cục bộ có tên rõ ràng. Nhưng cách làm của Kotlin là sử dụng các hàm gọn.

Các hàm gọn, hay còn gọi là hàm biểu thức đơn, là một mẫu phổ biến trong Kotlin. Khi một hàm trả về kết quả của một biểu thức duy nhất, bạn có thể chỉ định thân hàm sau dấu `=` , bỏ qua dấu ngoặc nhọn `{}`, và bỏ qua từ khóa `return`.

Trong tệp `Hello.kt`, thêm các hàm gọn để kiểm tra các điều kiện. Thay đổi hàm `shouldChangeWater()` để gọi các hàm mới. Chạy chương trình của bạn. Kết quả từ câu lệnh `println()` với `shouldChangeWater()` nên giống như trước khi bạn chuyển sang sử dụng các hàm gọn.

A screenshot of a computer program

Description automatically generated

Nhận xét: Output từ println() với shouldChangeWater() giống như trước khi chuyển sang sử dụng các hàm compact.

[5. Get started with filters](https://developer.android.com/codelabs/android-development-kotlin-2.1#4)

Step 1: Create a filter

Trong tệp `Hello.kt`, định nghĩa một danh sách các đồ trang trí bể cá ở cấp độ đầu tiên bằng cách sử dụng `listOf()`. Bạn có thể thay thế nội dung của `Hello.kt`.

Tạo một hàm `main()` mới với một dòng mã để in ra chỉ những đồ trang trí bắt đầu bằng chữ cái ‘p’. Mã cho điều kiện lọc nằm trong dấu ngoặc nhọn `{}`, và nó tham chiếu ngầm đến từng mục khi vòng lặp lọc qua danh sách. Nếu biểu thức trả về `true`, mục đó sẽ được bao gồm.

A screenshot of a computer program

Description automatically generated

Step 2: Compare eager and lazy filters

Danh sách kết quả được tạo ngay lập tức (eager) hay khi danh sách được truy cập (lazy)? Trong Kotlin, điều này xảy ra theo cách bạn cần. Theo mặc định, filter là eager, và mỗi lần bạn sử dụng filter, một danh sách mới được tạo ra.

Để làm cho bộ lọc lazy, bạn có thể sử dụng Sequence, là một tập hợp chỉ có thể xem một mục tại một thời điểm, bắt đầu từ đầu đến cuối. Thật tiện lợi, đây chính xác là API mà một bộ lọc lazy cần.

Trong tệp Hello.kt, thay đổi mã của bạn để gán danh sách đã lọc cho một biến gọi là eager, sau đó in ra biến đó. Dưới đoạn mã đó, đánh giá bộ lọc bằng cách sử dụng một Sequence với asSequence(). Gán chuỗi này cho một biến gọi là filtered, và in ra nó.

Khi bạn trả về kết quả lọc dưới dạng một Sequence, biến filtered sẽ không giữ một danh sách mới—nó sẽ giữ một Sequence các phần tử của danh sách và kiến thức về bộ lọc cần áp dụng cho các phần tử đó. Mỗi khi bạn truy cập các phần tử của Sequence, bộ lọc sẽ được áp dụng và kết quả sẽ được trả về cho bạn.

Buộc đánh giá của chuỗi bằng cách chuyển đổi nó thành một danh sách với toList(). In ra kết quả.

Để hình dung những gì đang xảy ra với Sequence và đánh giá lazy, hãy sử dụng hàm map(). Hàm map() thực hiện một phép biến đổi đơn giản trên mỗi phần tử trong chuỗi.

A screenshot of a computer program

Description automatically generated

In ra lazyMap, in phần tử đầu tiên của lazyMap bằng cách sử dụng first(), và in lazyMap đã chuyển đổi thành một danh sách.

Chạy chương trình và quan sát kết quả. Việc in lazyMap chỉ in ra tham chiếu đến Sequence—câu lệnh println() bên trong không được gọi. Việc in phần tử đầu tiên chỉ truy cập phần tử đầu tiên. Việc chuyển đổi Sequence thành một danh sách sẽ truy cập tất cả các phần tử.A screenshot of a computer program

Description automatically generated

Tạo một Sequence mới bằng cách sử dụng bộ lọc ban đầu trước khi áp dụng map. In ra kết quả đó.

Chạy chương trình và quan sát đầu ra bổ sung. Giống như việc lấy phần tử đầu tiên, câu lệnh println() bên trong chỉ được gọi cho các phần tử được truy cập.

A screenshot of a computer program

Description automatically generated

Một hàm biến đổi hữu ích khác cho các tập hợp trong Kotlin là flatten(). Hàm này tạo một danh sách từ một tập hợp của các tập hợp, ví dụ như một mảng của các mảng hoặc một danh sách của các danh sách.

Tạo một danh sách các danh sách. Sau đó, áp dụng hàm flatten() để biến tất cả các danh sách thành một danh sách duy nhất. In ra kết quả.

A screenshot of a computer program

Description automatically generated

[6. Get started with lambdas and higher-order functions](https://developer.android.com/codelabs/android-development-kotlin-2.1#5)

Lambda

Ngoài các hàm có tên truyền thống, Kotlin hỗ trợ lambdas tương tự như Java. Một lambda là một biểu thức mô tả một hàm. Nhưng thay vì khai báo một hàm có tên, bạn khai báo một hàm không có tên. Điều làm cho điều này hữu ích là biểu thức lambda giờ đây có thể được truyền dưới dạng dữ liệu. Trong các ngôn ngữ khác, lambdas thường được gọi là hàm ẩn danh, hàm literal, hoặc những tên tương tự.

Step 1: Learn about lambdas

Giống như các hàm có tên, lambdas cũng có thể có tham số. Đối với lambdas, các tham số (và kiểu của chúng, nếu cần) nằm bên trái của một ký hiệu gọi là mũi tên hàm ->. Mã thực thi nằm bên phải của mũi tên hàm. Khi lambda được gán cho một biến, bạn có thể gọi nó giống như một hàm.

Sử dụng REPL (Tools > Kotlin > Kotlin REPL), hãy thử đoạn mã sau:

A screenshot of a computer program

Description automatically generated

Trong ví dụ này, lambda nhận một tham số kiểu `Int` tên là `dirty`, và trả về `dirty / 2`.

Cú pháp của Kotlin cho các kiểu hàm có liên quan chặt chẽ đến cú pháp của nó cho lambdas. Sử dụng cú pháp này để khai báo một biến chứa một hàm một cách rõ ràng như sau:

Đây là những gì mã này nói:

1. Tạo một biến gọi là `waterFilter`.

2. `waterFilter` có thể là bất kỳ hàm nào nhận một `Int` và trả về một `Int`.

3. Gán một lambda cho `waterFilter`.

4. Lambda trả về giá trị của đối số `dirty` chia cho 2.

Step 2: Create a higher-order function

Một hàm bậc cao là một hàm nhận các hàm khác làm tham số, hoặc là một hàm trả về một hàm khác. Bạn có thể truyền một lambda vào một hàm bậc cao mà nhận một hàm làm đối số. Trong nhiệm vụ trước, bạn đã tạo ra một hàm bậc cao gọi là filter. Bạn đã truyền biểu thức lambda sau vào filter như là điều kiện để kiểm tra:

{ it[0] == 'p' }

Tương tự, map là một hàm bậc cao, và lambda mà bạn đã truyền cho nó là phép biến đổi để áp dụng.

Viết một hàm bậc cao. Đây là một ví dụ cơ bản, một hàm nhận hai tham số. Tham số đầu tiên là một số nguyên. Tham số thứ hai là một hàm nhận một số nguyên và trả về một số nguyên. Hãy thử nghiệm trong REPL.

Thân hàm sẽ gọi hàm được truyền vào như tham số thứ hai và truyền tham số đầu tiên cho nó.

Để gọi hàm này, hãy truyền vào một số nguyên và một hàm.

A screenshot of a computer program

Description automatically generated

Hàm bạn truyền vào không nhất thiết phải là một lambda; nó có thể là một hàm có tên thông thường. Để chỉ định đối số là một hàm thông thường, hãy sử dụng toán tử ::. Bằng cách này, Kotlin sẽ biết rằng bạn đang truyền tham chiếu đến hàm như một đối số, chứ không phải là cố gắng gọi hàm.

Hãy thử truyền một hàm có tên thông thường vào updateDirty(). Dưới đây là một ví dụ về cách thực hiện điều này:

A screen shot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

[7. Summary](https://developer.android.com/codelabs/android-development-kotlin-2.1#6)

* Để tạo các tệp nguồn Kotlin mới trong IntelliJ IDEA, nhấp vào "src" trong bảng Project và nhấp chuột phải để mở menu. Chọn "New -> Kotlin File/Class".
* Để biên dịch và chạy một chương trình trong IntelliJ IDEA, nhấp vào tam giác màu xanh lá bên cạnh hàm `main()`. Kết quả sẽ xuất hiện trong một cửa sổ bên dưới.
* Trong IntelliJ IDEA, bạn có thể chỉ định các đối số dòng lệnh để truyền vào hàm `main()` trong Run > Edit Configurations.
* Hầu như mọi thứ trong Kotlin đều có giá trị. Bạn có thể sử dụng điều này để làm cho mã của mình ngắn gọn hơn bằng cách sử dụng giá trị của một câu lệnh `if` hoặc `when` như một biểu thức hoặc giá trị trả về.
* Các đối số mặc định giúp loại bỏ nhu cầu tạo nhiều phiên bản của một hàm hoặc phương thức. Ví dụ: `fun swim(speed: String = "fast") { ... }`
* Các hàm ngắn gọn, hoặc hàm đơn biểu thức, có thể làm cho mã của bạn dễ đọc hơn. Ví dụ: `fun isTooHot(temperature: Int) = temperature > 30`
* Bạn đã học một số điều cơ bản về bộ lọc, sử dụng biểu thức lambda. Ví dụ: `val beginsWithP = decorations.filter { it[0] == 'p' }`
* Một biểu thức lambda là một hàm không bị ràng buộc với một định danh, tức là một hàm ẩn danh. Biểu thức lambda được định nghĩa trong dấu ngoặc nhọn {}.
* Trong một hàm bậc cao, bạn truyền một hàm, chẳng hạn như biểu thức lambda, vào một hàm khác dưới dạng dữ liệu. Ví dụ: `dirtyLevel = updateDirty(dirtyLevel) { dirtyLevel -> dirtyLevel + 23 }`

\_\_HẾT\_\_