Coroutine

A coroutine is similar to a thread (in the sense of multithreading): it is a line of execution, with its own stack, its own local variables, and its own instruction pointer; but it shares global variables and mostly anything else with other coroutines. The main difference between threads and coroutines is that, conceptually (or literally, in a multiprocessor machine), a program with threads runs several threads in parallel. Coroutines, on the other hand, are collaborative: at any given time, a program with coroutines is running only one of its coroutines, and this running coroutine suspends its execution only when it explicitly requests to be suspended.

Coroutines are a new way of writing asynchronous, non-blocking code (and much more)

Coroutine are light-weight threads. A light-weight thread means it doesn’t map on native thread, so it doesn’t require context switching on processor, so they are faster.

It doesn’t map on native thread

Coroutines and the threads both are multitasking. But the difference is that threads are managed by the OS and coroutines by the users.

Coroutines provide quasi-concurrent execution of program units (the coroutines) their execution is interleaved, but not overlapped:

Diagram

Description automatically generated

Diagram

Description automatically generated

Coroutines provide an easy way to manage threading and are perfectly suitable for processes execution on background threads when needed. This paradigm also provides an easy implementation of concurrency and structured concurrency. It is easily scalable and maintainable, which makes production cycle and efficiency of programmers’ time writing code, solving bugs or organizing and cleaning code better. Coroutines can significantly improve code performance and the ability to use it as a stream.

Rxjava

RxJava, which is the shorthand for “ReactiveX for Java” is a JVM library for doing asynchronous and executing event-based programs by using observable sequences. It's main building blocks are triple O's, Operator, Observer, and Observables. And using them we perform asynchronous tasks in our project. It makes multithreading very easy in our project. It helps us to decide on which thread we want to run the task.

It allows easy handling of cache without creating caching classes for instance. One of the most important things and a great advantage of RxJava is the decreasing of the memory leak in 90%, possible by replacing standard Android mechanisms, becomes really useful when you are developing Android applications that no longer need to take that into much consideration . It could optimize code to improve an application’s responsiveness and it’s very easy to scale yet, really hard to maintain due to the over usage of it.

Its’ complexity is challenging and the tendency to use it everywhere makes the code very complex, harder to debug and, consequently, making the programmer waste more time on it to solve bugs, develop new code around that, or even to maintain or rewrite code, since it doesn’t return errors and the only way to debug is quite primitive. This happens because the studio debugger cannot attach to a Rx chain and you need logs to see what is happening. Other point about this library: is not possible to emit null, although is not a great thing, since the probably of it emit null could be handled easily with errors.

In terms of performance, Coroutines is more efficient than RxJava as it uses less resources to execute the same task while also being faster in doing so. RxJava uses greater amounts of memory and requires more CPU time, which translates into higher battery consumption and possible UI interruptions for the user.

If the app runs network calls once or twice per activity creation, use Coroutines. If the app needs to be simple and easily read by less experienced coders, use Coroutines.

If the app runs any sort of real-time feature, use RxJava.

If the app needs high levels of data manipulation between fetching and emitting, use RxJava.

If the app has the architecture based on reactive stream, use RxJava.

If it is a multiplatform project based on Kotlin Native, use Coroutines.

It is possible to use both of them together, at least if the programming language is Kotlin.