Concurrency

Concurrent program characteristics:

Multiple tasks within a program which seem to be executed at the same time are said to be executed concurrently

Concurrent tasks can be completely independent, or they can have inter-dependencies which establish specific ordering rules between them

Concurrency is one of the most challenging topics in software and hardware development

Coroutines is a framework used to introduce concurrency into Kotlin applications

This course aims to provide the minimal amount of theory required to make an efficient use of Coroutines in your applications

Coroutines Intuition

Coroutine:

Coroutine is an abstract construct representing a sequence of steps.

Steps within a coroutine execute sequentially.

Steps within a coroutine can execute on different CoroutineDispatchers.

CoroutineDispatcher:

CoroutineDispatcher executes coroutines’ steps on a specific thread or group of threads.

Single coroutine can use multiple CoroutineDispatchers.

Dispatchers.Main is a special CoroutineDispatcher corresponding to UI thread in Android applications

Suspending Function:

Suspending function is a special type of function which can only be invoked within a coroutine (either directly, or by another suspending function).

Parent coroutine will wait for suspending function to complete before proceeding to the next step.

Suspending functions can “suspend” the execution, entering waiting state, but without blocking the calling thread.

Job:

Each coroutine is associated with a Job.

Coroutine’s Job is cancellable.

Cancellation of coroutine’s Job cancels the coroutine itself.

CoroutineScope:

CoroutineScope is kind of a “box” that contains launched coroutines.

More than one coroutine can be associated with CoroutineScope at any instant of time.

CoroutineScope can cancel all its children coroutines.

With your new intuition, you can use Coroutines to implement simple concurrent flows

… but I wouldn’t recommend that

Topics we haven’t covered yet:

Detailed cancellation mechanics.

Error handling.

Parallel decomposition.

Structured Concurrency.

More…

We will cover all these topics in the next lectures!