What is a Thread?

The basic component of every executable program is a set of instructions. These instructions are often dependent on each other and run sequentially one after the other once the program is scheduled for execution by the CPU scheduler. ***A Thread is a set of instructions that can be managed independently by the scheduler***. Multiple threads can exist simultaneously within a program/process, executing concurrently and sharing the resources of the system.

Multithreading

Multithreading is a program execution model that allows multiple threads to co-exist within the context of a program and perform various tasks while executing independent of each other. In a multiprocessing system (i.e. a system with multiple CPUs), multithreading can be used to achieve parallel execution. ***One of the biggest advantages of a multithreading system in Android is responsiveness***. Multithreading allows the system to remain responsive to user interactions and other events while simultaneously processing data. Without multithreading, an application may appear to freeze while it fetches data from the background or performs any heavy computation.

Runnable Interface

The runnable interface is an interface that ***provides a common protocol for the objects that wish to run a code segment when they are active***, such as threads. This interface contains a single function that contains the executable code. ***When the thread is active, the run() function of the Runnable interface is called***.

Threads in Android

The Android system creates a UI Thread (or the main thread) when the application is launched. This thread is responsible for all the tasks that take place within the android application, from updating the UI, to responding to system calls and address events that occur within the system. ***All of the app instructions are run on the main thread by default*** and heavy task instructions can render the app unresponsive, making multithreading a necessity for applications with heavy processing.

How to create Threads

Kotlin has a Thread class that can be used to create the objects of Threads. The Thread class has an object of the Runnable interface as its constructor parameter. An object of the thread class can be created as follows:

```
val thread = Thread(object: Runnable{
      override fun run() {
            //code for the Thread
      }
})
```

Since, the Runnable is an interface with a single function, it can be replaced with a lambda function. This can be done as follows:

```
val thread = Thread {
      //code for the Thread
}
```

You will learn more about threads and parallel (or asynchronous) processing in the following lecture.

To read more about threads and processes in Android please visit:

https://developer.android.com/guide/components/processes-and-threads