**Multithreading**

More than one thread of execution within a process. This is called multi-threading.

**Thread**

Thread is a lightweight sub-process that provides us a way to do background operations without interrupting the User Interface (UI).

When an app is launched, it creates a single thread in which all app components will run by default.

The thread which is created by the runtime system is known as the **main thread** or **UI thread**.

The main thread is nothing but a handler thread. The main thread is responsible for handling events from all over the app.

**It's widely known that it's illegal to update UI components directly from threads other than main thread in android.**

**Handler**

A Handler allows you to send and process [Message](https://developer.android.com/reference/android/os/Message) and Runnable objects associated with a thread's [MessageQueue](https://developer.android.com/reference/android/os/MessageQueue). Each Handler instance is associated with a single thread and that thread's message queue.

When you create a new Handler, it is bound to a [Looper](https://developer.android.com/reference/android/os/Looper). It will deliver messages and Runnables to that Looper's message queue and execute them on that Looper's thread.

The Runnable r to be added to the message queue. The runnable will be run on the thread to which this handler is attached.

public final boolean post ([Runnable](https://developer.android.com/reference/java/lang/Runnable) r)

**r** : The Runnable that will be executed. This value cannot be null.

Boolean :Returns true if the Runnable was successfully placed in to the message queue. Returns false on failure.

**Looper**

Class used to run a message loop for a thread.

Threads by default do not have a message loop associated with them; to create one, call [**prepare()**](https://developer.android.com/reference/android/os/Looper#prepare()) in the thread that is to run the loop, and then [**loop()**](https://developer.android.com/reference/android/os/Looper#loop()) to have it process messages until the loop is stopped.

Most interaction with a message loop is through the [Handler](https://developer.android.com/reference/android/os/Handler) class.

[prepare](https://developer.android.com/reference/android/os/Looper#prepare())() **:** Initialize the current thread as a looper.

[loop](https://developer.android.com/reference/android/os/Looper#loop())() **:**  Run the message queue in this thread

[quit](https://developer.android.com/reference/android/os/Looper#quit())() **:** Quits the looper.

**Message Queue**

Low-level class holding the list of messages to be dispatched by a [Looper](https://developer.android.com/reference/android/os/Looper).

Messages are not added directly to a MessageQueue, but rather through [Handler](https://developer.android.com/reference/android/os/Handler) objects associated with the Looper.

**HandlerThread**

It is an extension of Thread, which works with a Looper. Meaning, it is meant to handle multiple jobs on the background thread.