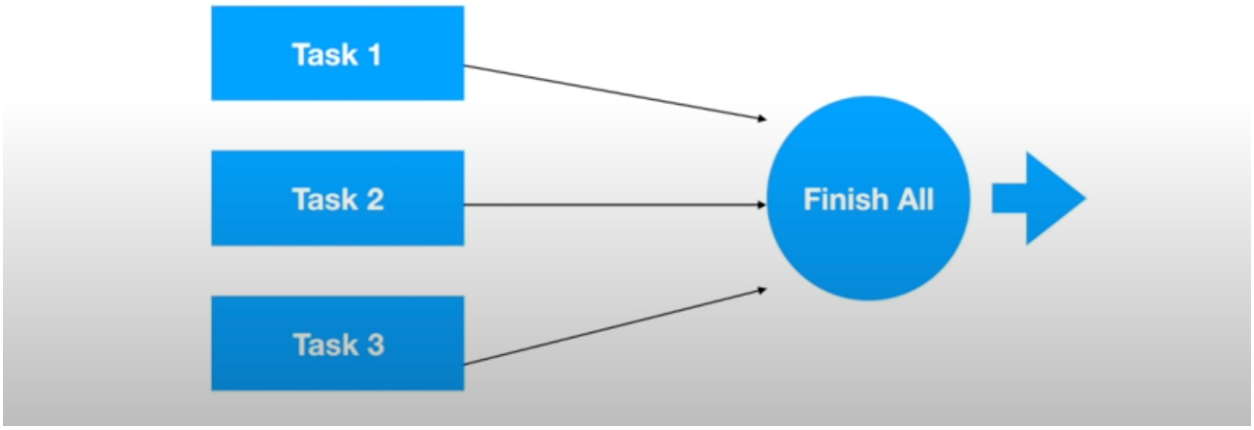
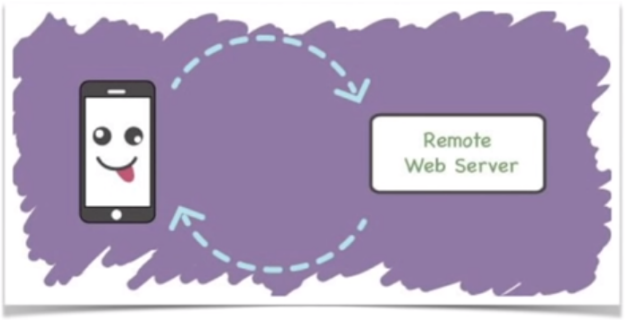
What we are going to discuss today?

- We will write a Java program that execute several tasks in parallel and then send a callback when all of those tasks completes.



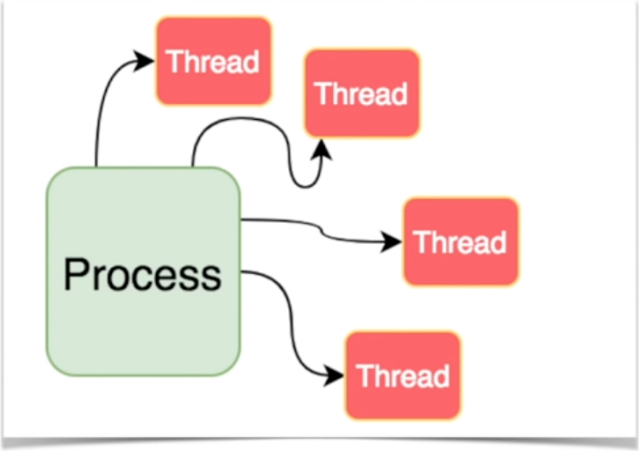
Some Use Cases

- **Android:** Synchronize app's data with the server through APIs and remove the loader when all completes.
- **Backend:** Save data in database, file logs, and cache. Send response on all the task completion.



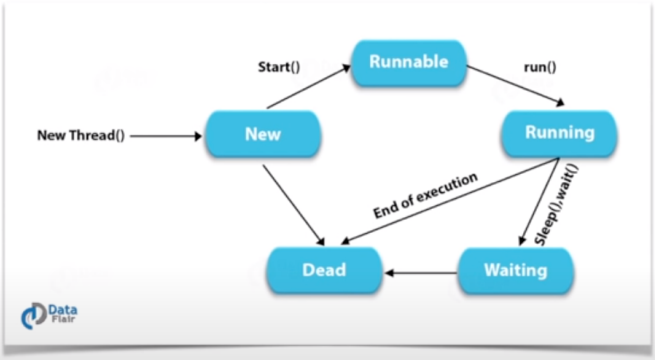
Process

- A process is an instance of a computer program that is being executed.
- When an Android application starts, the Android system starts a new Linux process for the application with a single thread of execution called the main thread.



Threads

- A program can contains two or more parts that can run in parallel. Each part of such a program is called a thread.
- They help in utilizing the multicore processors and also reduce the ideal CPU time of a single processor.
- Creating too many threads slows down the execution
- Only a few threads run in parallel, others wait for the CPU to get free.



Creating Java Threads

Thread subclass

```
public class MyThread extends Thread {

    public void run(){
       System.out.println("MyThread running");
    }
  }

MyThread myThread = new MyThread();
myTread.start();
```

Implement Runnable

```
public class MyRunnable implements Runnable {

    public void run(){
       System.out.println("MyRunnable running");
    }
  }

Thread thread = new Thread(new MyRunnable());
thread.start();
```

Program Components

- **Task**: It is a Runnable that will contains the code to be executed in a separate thread.
- **Worker**: It will be responsible for creating a thread and running the supplied task.
- **Executor**: It will create workers to handle the tasks. It will also be responsible for broadcasting the completion of all the tasks.

Next Devlog

- **Callables**, **Future** and **FutureTask**: They help in passing the result to the caller thread when the worker thread finishes.
- **ThreadPoolExecutor**: Execute large number of tasks using a pool of threads.
- **Monitor** and **Locks**