Concurrency in Java

I skipped the theory part in the beginning of the lecture 17 of Java Cup series.

# How to Create New Threads

You can either

* Extend the Thread class and override its run() method and call .start() on its instances

Or

* Implement the Runnable interface and feed it as a target to new instance of the Thread class and then call the .start() method on that instance

How it works:

By default, the Thread class in its run method calls the run method of its target which is of type runnable. If the target is null then it’ll return with no action. So, you need either to override this method, or feed the Thread instance with an implementation of Runnable.

* The run() method in Runnable and Thread defines the codes that will run in the new thread, it doesn’t create a new thread itself.
* The start() method is a low-level method that creates a new thread and calls the run() method.

The interface approach is preferable and is most commonly implemented since:

* You can extend another class when you are implementing Runnable but you can not when you extend Thread
* I guess: Other approaches and tools to create threads and shit are based on this approach

**So, no matter the approach there will be a new instance of the Thread class when creating a new thread whose start() method will be called.**

## Other Methods In the Thread class

* **Instance methods:**
* We talked about run() and start()
* **Join():**

when you call this method on a thread instance, the current thread that’s invoking this method will stop execution until the thread on which the join method has been invoked has finished

* setPriority()
* setDaemon()
* getId()
* Static Methods
  + Thread.currentThread(): returns the current thread that its code is running right now
  + Thread.sleep(m, n): **the current thread** will stop running for m milliseconds and n nanoseconds