Difference between Runnable and Callable

What's the difference between Runnable and Callable?

Runnable's Functional Method	**Callable's Functional Method**
void run()	V call() throws Exception

Significantly, Callable returns a value.

This means you can get data back from your running threads.

execute vs. submit

On this slide, I'm showing you the method signatures for both the execute and submit methods.

method	**signature**
execute	void execute(Runnable command)
submit	Future<?> submit(Runnable task) <T> Future<T> submit(Runnable task, T result) <T> Future<T> submit(Callable<T> task)

The Future Interface

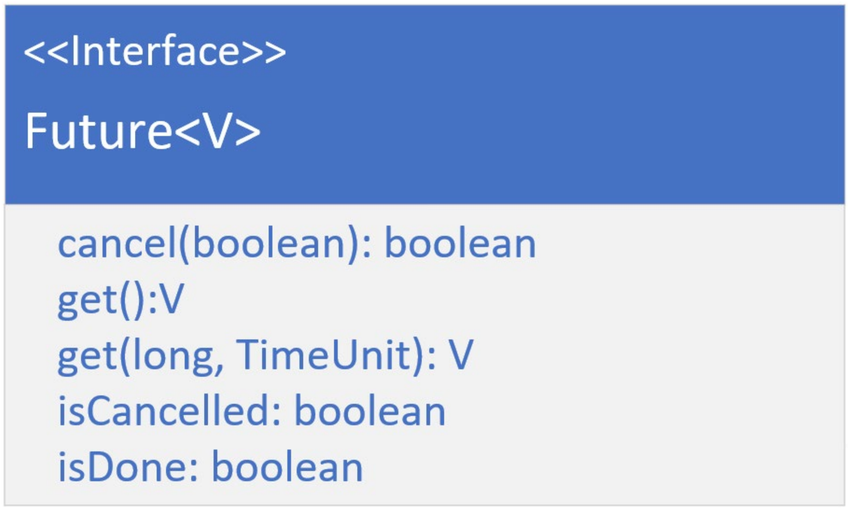
A Future represents a result, of an asynchronous computation.

It's a generic type, a placeholder for a result instance.

It has methods that cancel the task, retrieve the result, or check if the computation was completed or cancelled.

The get method returns the result, but you can only call this get method, when the computation is complete, otherwise the call will block, until it does complete.

The overloaded version of the get method allows you to specify a wait time, rather than blocking.



invokeAll vs. invokeAny

When would you want to use invoke Any vs. invokeAll?

Characteristic	invokeAny	invokeAll
Tasks Executed	At least one, the first to complete.	All tasks get executed.
Result	Result of the first task to complete, not a Future.	Returns a list of results, as futures, for all of the tasks, once they have all completed.
Use cases	Use this method when you need a quick response back from one of several tasks, and you don't care if some will fail.	Use this method when you want all the tasks to be executed concurrently, and all tasks must complete before proceeding.