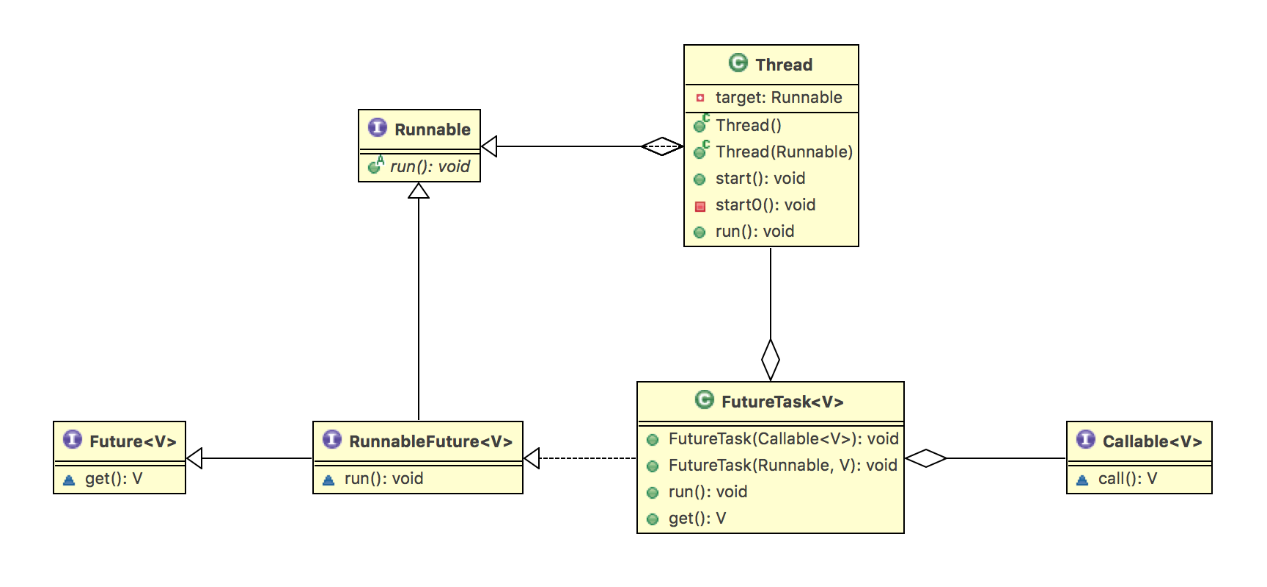
线程启动原理

Java多线程，皆始于Thread。Thread是多线程的根，每一个线程的开始都始于Thread的start()方法。那么线程是如何被开启，run方法是如何被执行的呢？



这张图在今后的几个章节都会用到，其中只展示了部分关键方法。

我们都知道启动一个线程必须调用一个Thread的start()方法。在面试时经常可能会被问到start()和run()方法的区别，为什么一定要用start()方法才是启动线程？对比start()方法和run()的源码一看便知：

/\*\*

\* Causes this thread to begin execution; the Java Virtual Machine calls the <code>run</code> method of this thread.

\* 1、start方法将导致this thread开始执行。由JVM调用this thread的run方法。

\* The result is that two threads are running concurrently: the current thread (which returns from the call to the <code>start</code> method) and the other thread (which executes its <code>run</code> method).

\* 2、结果是 调用start方法的当前线程 和 执行run方法的另一个线程 同时运行。

\* It is never legal to start a thread more than once.In particular, a thread may not be restarted once it has completed execution.

\* 3、多次启动线程永远不合法。 特别是，线程一旦完成执行就不会重新启动。

\* @exception IllegalThreadStateException if the thread was already started.

\* 如果线程已启动，则抛出异常。

\* @see #run()

\* @see #stop()

\*/

public synchronized void start() {

/\*\*

\* This method is not invoked for the main method thread or "system" group threads created/set up by the VM. Any new functionality added to this method in the future may have to also be added to the VM.

\* 4、对于由VM创建/设置的main方法线程或“system”组线程，不会调用此方法。 未来添加到此方法的任何新功能可能也必须添加到VM中。

\* A zero status value corresponds to state "NEW".

\* 5、status=0 代表是 status 是 "NEW"。

\*/

if (threadStatus != 0)

throw new IllegalThreadStateException();

/\* Notify the group that this thread is about to be started so that it can be added to the group's list of threads and the group's unstarted count can be decremented.

\* 6、通知组该线程即将启动，以便将其添加到线程组的列表中，并且减少线程组的未启动线程数递减。

\* \*/

group.add(this);

boolean started = false;

try {

//7、调用native方法，底层开启异步线程，并调用run方法。

start0();

started = true;

} finally {

try {

if (!started) {group.threadStartFailed(this);}

} catch (Throwable ignore) {

/\* do nothing. If start0 threw a Throwable then it will be passed up the call stack

\* 8、忽略异常。 如果start0抛出一个Throwable，它将被传递给调用堆栈。

\*/

}

}

}

//native方法，JVM创建并启动线程，并调用run方法

private native void start0();

对于源码中的注释并没有省略，都进行了翻译，可以更好的理解整个启动过程。其中有几个需要注意的点：

start方法用synchronized修饰，为同步方法；

虽然为同步方法，但不能避免多次调用问题，用threadStatus来记录线程状态，如果线程被多次start会抛出异常；threadStatus的状态由JVM控制。

使用Runnable时，主线程无法捕获子线程中的异常状态。线程的异常，应在线程内部解决。

看完start()方法，线程的启动逻辑已经比较清楚，要探究更底层的原理就需要探究native方法start0()了。