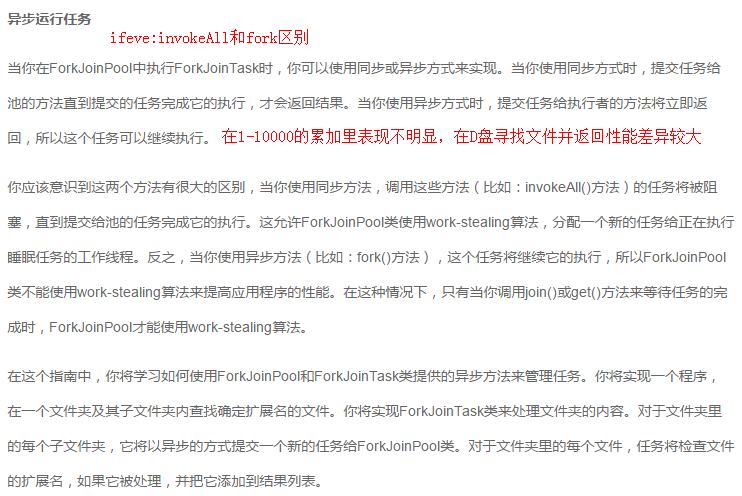
**abstract** **class** RecursiveTask<V> **extends** ForkJoinTask<V> 有返回值

**abstract** **class** RecursiveAction<V> **extends** ForkJoinTask<V> 无返回值

invokeAll()和fork()都是ForkJoinTask的抽象方法，可以在实现类的compute()方法里直接调用



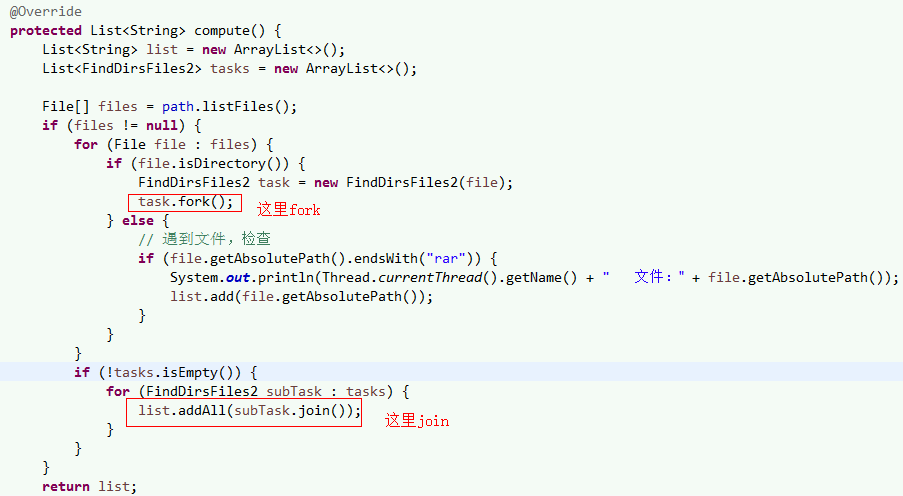
invokeAll()和fork()使用上的区别

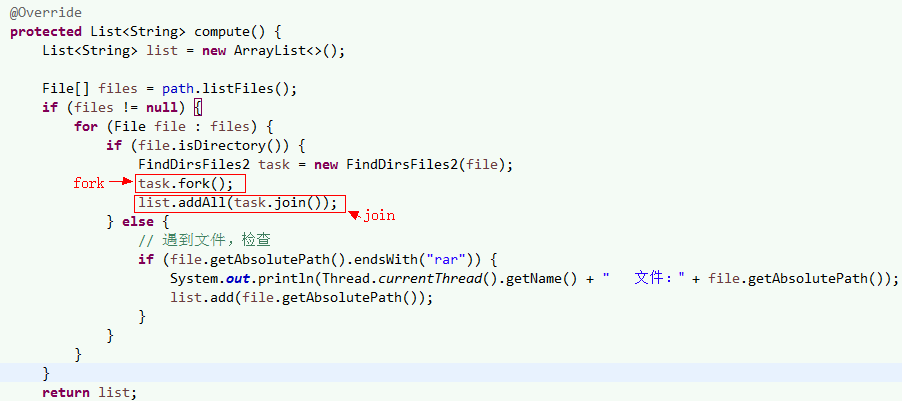




在for循环里都调用了join()方法，因为compute要返回最终的list结果，所以要阻塞线程将所有的值合并到一个集合里面。fork()两种实现方式：

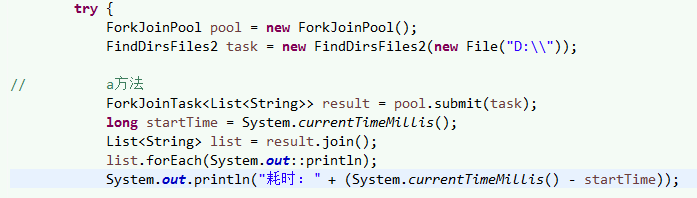
基本上采用invokeAll()方法，

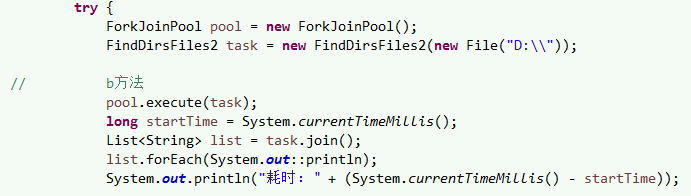




a和b的区别：

a返回的是顶级父类，再调用join()和get()；b直接通过实现类调用join()和get()





ForkJoinPool pool = ForkJoinPool.commonPool();

ForkJoinPools类有一个静态方法commonPool()， 这个静态方法所获得ForkJoinPools实例是由整个应用进程共享的，并且它适合绝大多数的应用系统场景。 使用commonPool通常可以帮助应用程序中多种需要进行归并计算的任务共享计算资源，从而使后者发挥最大作用（ForkJoinPools中的工作线程在闲置时会被缓慢回收，并在随后需要使用时被恢复）