# Callable & Future

# 获取线程的计算结果

- 共享变量
- Callable & Future
- CompletableFuture　　Stream
- ForkJoin框架　　ForkJoinPool  ForkJoinTask

# Callable vs. Runnable

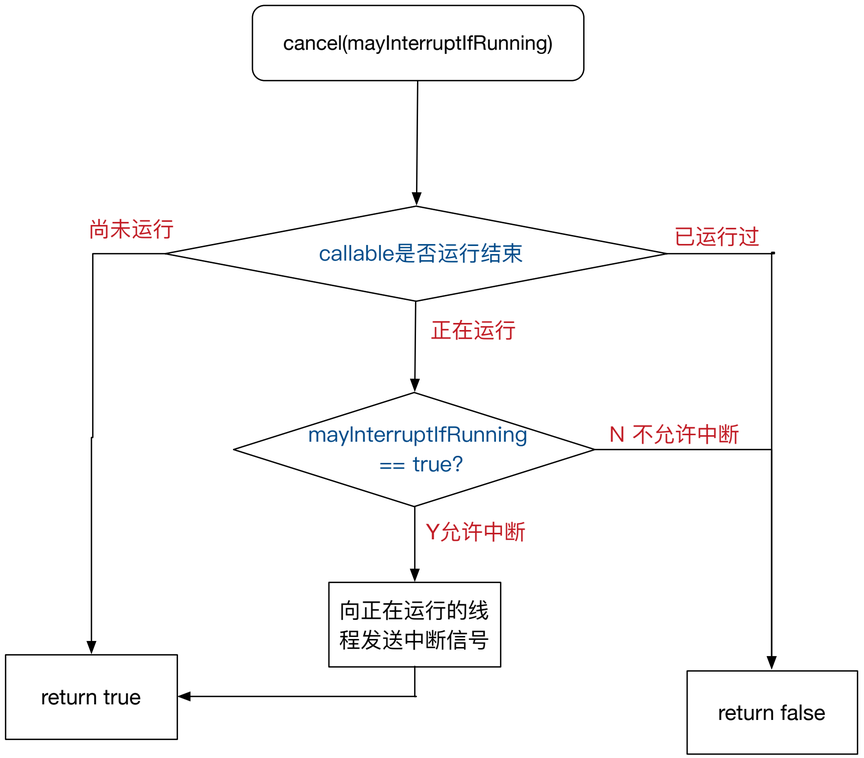
| java.util.concurrent.Callable<V> | java.lang.Runnable |
| --- | --- |
| V call() | void  run() |
| ExecutorService service = Executors.*newCachedThreadPool*(); Future<Object> future =  service.submit(callable); // 可以执行其他代码，*callable*会自动开始计算并将结果保存到*Future*内 Object result = future.get(); //如果*callable*的*call*方法尚未运算完毕，会等待运算完毕 | new Thread(runnable).start() |

```
FetureTask<Object> task = new FetureTask<>(runnable);
new Thread(task).start();
…
Object result = task.get();
```

# Callable

- Callable
  - V call() throws Exception;
- Runnable
  - void run();

# Future接口




- boolean cancel(boolean mayInterruptIfRunning)
- boolean isCancelled()
- boolean isDone() // 正常结束、抛出异常、被取消都返回true
- V get() throws InterruptedException, ExecutionException
- V get(long timeout, TimeUnit unit)
  throws InterruptedException, ExecutionException, TimeoutException

# 异步运行Callable （一）

```
Callable<Object> callable = …;

ExecutorService threadPool = Executors.newCachedThreadPool();

Future<Object> future = threadPool.submit(callable);

Object result = future.get();
```

# 异步运行Callable （二）

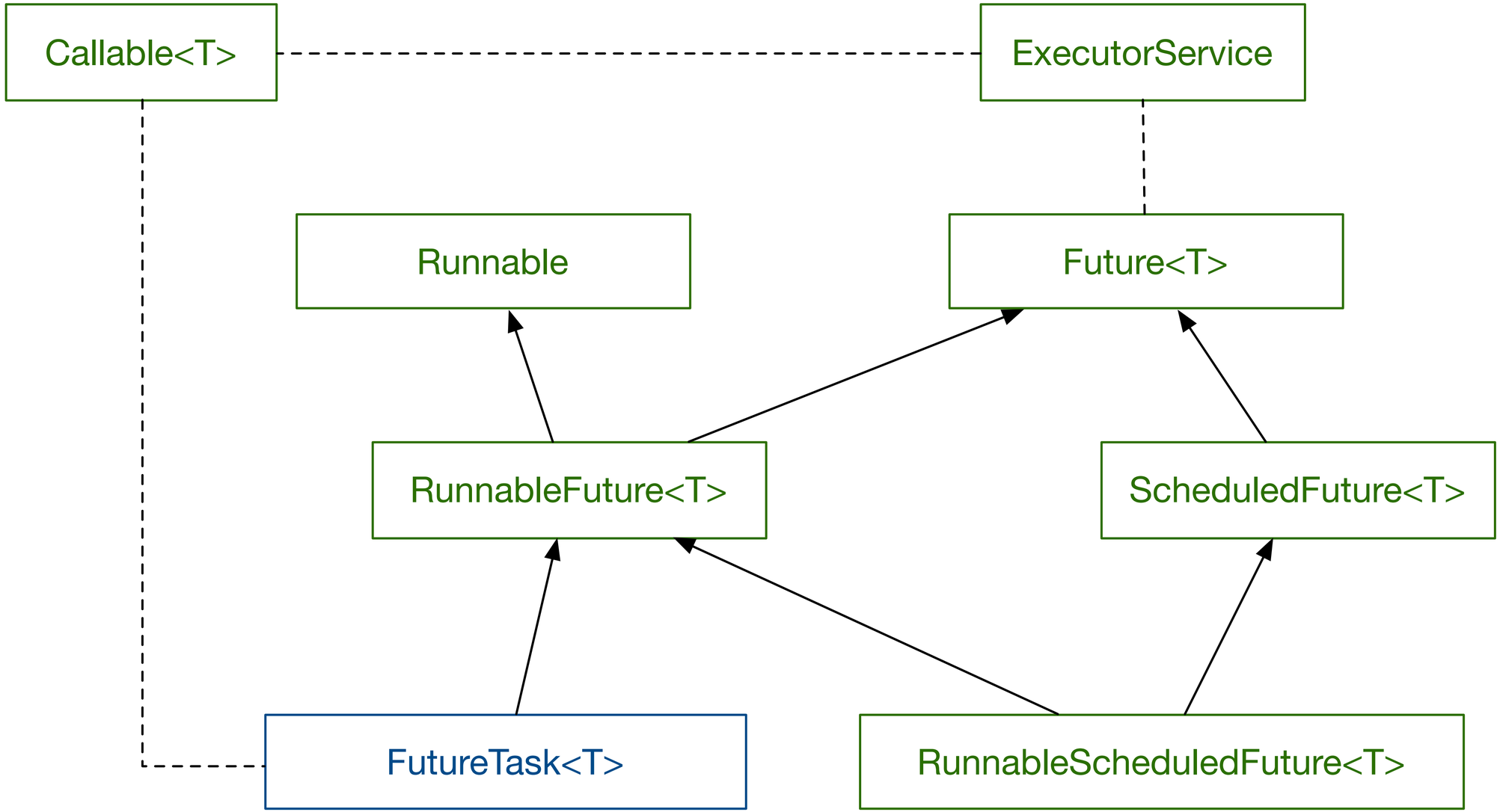
```
Callable<Object> callable = …;

FutureTask<String> task = new FutureTask(callable);

(new Thread(task)).start();

Object result = task.get();
```

```
try{
   Object result = task.get(1, TimeUnit.MINUTES);
}catch(TimeoutException the){
   // 超时没得到结果
}
```


Callable<T>
ExecutorService
Runnable
Future<T>
RunnableFuture<T>
ScheduledFuture<T>
FutureTask<T>
RunnableScheduledFuture<T>