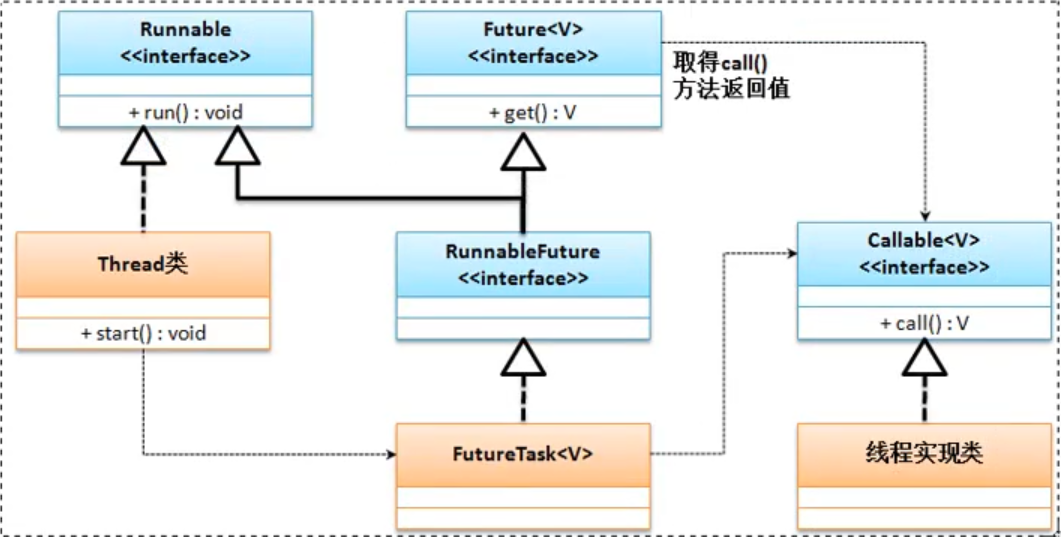
1. **Callable接口实现多线程**

最传统的多线程依靠的是Runnable，但是Runnable有一个缺点：当线程执行完毕之后无法获取返回值。所以从JDK1.5之后，我们提出一个新的线程实现接口:java.util.concurrent.Callable接口。首先我们观察这个接口的定义：

|  |
| --- |
| @FunctionalInterface  public interface Callable<V>{  V call() throws Exception  } |

可以发现Callable定义的时候可以设置一个泛型，此泛型的类型就是返回数据的类型，这样的好处就是可以避免向下转型所带了的安全隐患。



范例:使用callable实现多线程

|  |
| --- |
| package com.company;  import java.util.concurrent.Callable ;  import java.util.concurrent.ExecutionException;  import java.util.concurrent.Future;  import java.util.concurrent.FutureTask;  /\*\*  \* @author JackWen  \*/  public class CallableMain<S> {  public static void main(String[] args) throws ExecutionException, InterruptedException {  FutureTask<String> task = new FutureTask<String>(new MyThread2()) ;  new Thread(task).start();  System.out.println("线程返回数据:" + task.get());  }  }  class MyThread2 implements Callable<String>{  @Override  public String call() throws Exception {  for (int i = 0; i < 10; i++) {  System.out.println("线程执行");  }  return "线程执行完毕";  }  } |

Runnable与Callable的区别：

Runnable是在1.0提出，而Callable是在15提出

Runnable无返回值，Callable有返回值（泛型写上面类型就返回上面类型）

Runnbale的方法为run ，Callable的方法为Call

Runnable不会抛出异常，而Callable会抛出异常。