1.callable规定的方法是call()，Runnable规定的方法是run()。
2.callable的任务执行后可返回值，而Runnable的任务是不能返回值的。
3.call方法可以抛出异常，run方法不可以。
4.运行callable任务可以拿到一个Future对象，Future表示异步计算结果。它提供了检查计算是否完成的方法，以等待计算的完成，并获取计算的结果。计算完成后只能使用get方法来获取结果，如果线程没有完成，Future.get()方法可能会阻塞当前线程的执行；如果线程出现异常或者取消，会抛出异常.isDone确定任务是正常完成了还是被取消了，一旦计算完成，就不能再取消计算。

结合JDK的Future来看，就是你run线程后，你可以把线程的返回值赋给Future并返回一个Future对象。这时你可以立即拿到这个对象，然后进行下面的逻辑。但是如果你要get这个Future中的线程结果，就会被阻塞直到线程结束。
就相当于现在的期房，你把手续和钱都交上去了，就可以马上拿到合同，但只有合同没有房子。这个时候你已经是有房一族了，你可以先去买家电买装修（走下面的其他逻辑）。但是你要把家电和装修放进去，就必须等到房子完工（阻塞）。

```
import java.util.concurrent.*;
public class Test {
    public static void main(String[] args) throws InterruptedException,
            ExecutionException {
        final ExecutorService exec = Executors.newFixedThreadPool(5);
        Callable<String> call = new Callable<String>() {
            public String call() throws Exception {
                Thread.sleep(1000 * 10);//休眠指定的时间，此处表示该操作比较耗时
                return "Other less important but longtime things.";
            }
        };
        Future<String> task = exec.submit(call);
        //重要的事情
        System.out.println("Let's do important things. start");
        Thread.sleep(1000 * 3);
        System.out.println("Let's do important things. end");
```

```
        //不重要的事情
        while(! task.isDone()){
            System.out.println("still waiting....");
            Thread.sleep(1000 * 1);
        }
        System.out.println("get sth....");
        String obj = task.get();
        System.out.println(obj);
        //关闭线程池

        exec.shutdown();
    }
}
```

```
Let's do important things. start
Let's do important things. end
still waiting
still waiting
still waiting
still waiting
still waiting
still waiting
still waiting
get sth...
other less important but longTime things
```