java程序运行至少启动2个线程：main线程和垃圾收集线程。因为每当使用java命令执行一个类的时候，实际上都会启动一个JVM，每一个JVM实际在就是在操作系统中启动了一个进程。

通过Runnable接口实现多线程

```
public class TestThread2 implements Runnable {//自定义类实现Runnable接口；
    //需实现run()方法，run()方法里是线程体；
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println(Thread.currentThread().getName() + ":" + i
);
        }
    }
    public static void main(String[] args) {
        //创建线程对象，把实现了Runnable接口的对象作为参数传入；
        Thread thread1 = new Thread(new TestThread2());
        thread1.start();//启动线程；
        Thread thread2 = new Thread(new TestThread2());
        thread2.start();
    }
}
```

Callable接口

与 Runnable 相比，Callable 可以有返回值，返回值通过 FutureTask 进行封装。

```
public class MyCallable implements Callable<Integer> {
    public Integer call() {
        return 123;
    }
}
public static void main(String[] args) throws ExecutionException, InterruptedException
{
    MyCallable mc = new MyCallable();
    FutureTask<Integer> ft = new FutureTask<>(mc);
    Thread thread = new Thread(ft);
    thread.start();
    System.out.println(ft.get());
}
```

特点：

1）：适合多个相同的程序代码的线程去处理同一个资源

2）：可以避免java中的单继承的限制

3）：类可能只要求可执行就行，继承整个 Thread 类开销过大。

4）：线程池只能放入实现Runable或callable类线程，不能直接放入继承Thread的类