Java通过Executors提供四种线程池，分别为：

newCachedThreadPool创建一个可缓存线程池，如果线程池长度超过处理需要，可灵活回收空闲线程，若无可回收，则新建线程。

newFixedThreadPool 创建一个定长线程池，可控制线程最大并发数，超出的线程会在队列中等待。

newScheduledThreadPool 创建一个定长线程池，支持定时及周期性任务执行。

newSingleThreadExecutor 创建一个单线程化的线程池，它只会用唯一的工作线程来执行任务，保证所有任务按照指定顺序(FIFO, LIFO, 优先级)执行。

优点

重用存在的线程，减少对象创建、消亡的开销，性能佳。

可有效控制最大并发线程数，提高系统资源的使用率，同时避免过多资源竞争，避免堵塞。

提供定时执行、定期执行、单线程、并发数控制等功能。

**1、继承Thread类：**

　　　　步骤：①、定义类继承Thread；

　　　　　②、复写Thread类中的run方法；

　　　　目的：将自定义代码存储在run方法，让线程运行

　　　　　③、调用线程的start方法：

　　　　该方法有两步：启动线程，调用run方法。

**2、实现Runnable接口：** 接口应该由那些打算通过某一线程执行其实例的类来实现。类必须定义一个称为run 的无参方法。

 　　　　实现步骤：  ①、定义类实现Runnable接口

　　　　　　　　　　②、覆盖Runnable接口中的run方法

　　　　　　　　　　　　　将线程要运行的代码放在该run方法中。

　　　　　　　　　　③、通过Thread类建立线程对象。

　　　　　　　　　　④、将Runnable接口的子类对象作为实际参数传递给Thread类的构造函数。

　　　　　　　　　　　　　自定义的run方法所属的对象是Runnable接口的子类对象。所以要让线程执行指定对象的run方法就要先明确run方法所属对象

　　　　　　　　　　⑤、调用Thread类的start方法开启线程并调用Runnable接口子类的run方法。

**3、通过Callable和Future创建线程：**

　　　　实现步骤：①、创建Callable接口的实现类，并实现call()方法，改方法将作为线程执行体，且具有返回值。

　　　　　　　　　②、创建Callable实现类的实例，使用FutrueTask类进行包装Callable对象，FutureTask对象封装了Callable对象的call()方法的返回值

　　　　　　　　　③、使用FutureTask对象作为Thread对象启动新线程。

　　　　　　　　　④、调用FutureTask对象的get（）方法获取子线程执行结束后的返回值。



