1）继承Thread类创建线程

2）实现Runnable接口创建线程

3）使用Callable和FutureTask创建线程

4）使用线程池例如用Executor框架

第一种:

1】定义Thread类的子类，并重写该类的run()方法，该方法的方法体就是线程需要完成的任务，run()方法也称为线程执行体。

2】创建Thread子类的实例，也就是创建了线程对象

3】启动线程，即调用线程的start()方法

第二种:

1】定义Runnable接口的实现类，一样要重写run()方法，这个run（）方法和Thread中的run()方法一样是线程的执行体

2】创建Runnable实现类的实例，并用这个实例作为Thread的target来创建Thread对象，这个Thread对象才是真正的线程对象

3】第三部依然是通过调用线程对象的start()方法来启动线程

第三种:

1】创建Callable接口的实现类，并实现call()方法，然后创建该实现类的实例（从java8开始可以直接使用Lambda表达式创建Callable对象）。

```
FutureTask<Integer> future = new FutureTask<Integer>(new
Callable<Integer>() {
    public Integer call() throws Exception {
        return 5;
    }
});
```

2】使用FutureTask类来包装Callable对象，该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对象的target创建并启动线程（因为FutureTask实现了Runnable接口）

```
new Thread(future, "有返回值的线程").start();
```

4】调用FutureTask对象的get()方法来获得子线程执行结束后的返回值

第四种:

1. 创建线程池对象。

```
ExecutorService service = Executors.newFixedThreadPool(2);
```

2. 创建Runnable接口子类对象,并重写run()方法

```
Runnable runnable=new Runnable() {
    public void run() {
        System.out.println("哈哈");
    }
};
```

3. 线程池对象调用submit(Runnable接口子类对象)方法

```
service .submit(runnable);
```