### 1、继承Thread类创建线程类

* 继承Thread类，并重写run方法，该run方法的方法体就代表了线程要完成的任务。因此run()方法称为执行体。
* 创建Thread子类的实例，即创建了线程对象。
* 调用线程对象的start()方法来启动该线程。

Thread.currentThread()方法返回当前正在执行的线程对象。getName()方法返回调用该方法的线程的名字。

### 2、通过Runnable接口创建线程类

* 定义runnable接口的实现类，并重写该接口的run()方法，该run()方法的方法体同样是该线程的线程执行体。
* 创建 Runnable实现类的实例，并以此实例作为Thread的target来创建Thread对象，该Thread对象才是真正的线程对象。
* 调用线程对象的start()方法来启动该线程。

线程的执行流程很简单，当执行代码start()时，就会执行对象中重写的void run();方法，该方法执行完成后，线程就消亡了。

### 3、通过Callable和FutureTask创建线程

1. 创建Callable接口的实现类，并实现call()方法，该call()方法将作为线程执行体，并且有返回值。
2. 创建Callable实现类的实例，使用FutureTask类来包装Callable对象，该FutureTask对象封装了该Callable对象的call()方法的返回值。（FutureTask是一个包装器，它通过接受Callable来创建，它同时实现了Future和Runnable接口。）
3. 使用FutureTask对象作为Thread对象的target创建并启动新线程。
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

### 4、比较

#### 4.1 采用实现Runnable、Callable接口的方式创建多线程时

优势是：

* 线程类只是实现了Runnable接口或Callable接口，还可以继承其他类。
* 在这种方式下，多个线程可以共享同一个target对象，所以非常适合多个相同线程来处理同一份资源的情况，从而可以将CPU、代码和数据分开，形成清晰的模型，较好地体现了面向对象的思想。

劣势是：

* 编程稍微复杂，如果要访问当前线程，则必须使用Thread.currentThread()方法。

#### 4.2 使用继承Thread类的方式创建多线程时

优势是：

* 编写简单，如果需要访问当前线程，则无需使用Thread.currentThread()方法，直接使用this即可获得当前线程。

劣势是：

* 线程类已经继承了Thread类，所以不能再继承其他父类。

#### 4.3 Runnable和Callable的区别

1. Callable规定（重写）的方法是call()，Runnable规定（重写）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，表示异步计算的结果。它提供了检查计算是否完成的方法，以等待计算的完成，并检索计算的结果。通过Future对象可以了解任务执行情况，可取消任务的执行，还可获取执行结果。