一、线程的三种实现方式

1. 继承Thread类
2. 实现runnable接口
3. **实**现Callable接口通过FutureTask包装器来创建Thread线程
   1. <http://www.cnblogs.com/dolphin0520/p/3949310.html>
4. 使用ExecutorService、Callable、Future实现有返回结果的线程
5. 参考文献
   1. <https://www.cnblogs.com/felixzh/p/6036074.html>

二、区别和联系

1. Runnable和callable
   1. 相同点：
      1. 两者都是接口；（废话）
      2. 两者都可用来编写多线程程序；
      3. 两者都需要调用Thread.start()启动线程；
   2. 不同点：
      1. 两者最大的不同点是：实现Callable接口的任务线程能返回执行结果；而实现Runnable接口的任务线程不能返回结果；
      2. Callable接口的call()方法允许抛出异常；而Runnable接口的run()方法的异常只能在内部消化，不能继续上抛；
   3. 注意点：
      1. Callable接口支持返回执行结果，此时需要调用FutureTask.get()方法实现，此方法会阻塞主线程直到获取‘将来’结果；当不调用此方法时，主线程不会阻塞！
   4. 参考文献
      1. <https://www.cnblogs.com/frinder6/p/5507082.html>
2. Runnable和thread
   1. 区别
      1. 适合多个相同程序代码的线程去处理同一资源的情况，把虚拟CPU（线程）同程序的代码，数据有效的分离，较好地体现了面向对象的设计思想。
      2. 可以避免由于Java的单继承特性带来的局限。我们经常碰到这样一种情况，即当我们要将已经继承了某一个类的子类放入多线程中，由于一个类不能同时有两个父类，所以不能用继承Thread类的方式，那么，这个类就只能采用实现Runnable接口的方式了。
      3. 有利于程序的健壮性，代码能够被多个线程共享，代码与数据是独立的。当多个线程的执行代码来自同一个类的实例时，即称它们共享相同的代码。 多个线程操作相同的数据，与它们的代码无关。当共享访问相同的对象是，即它们共享相同的数据。当线程被构造时，需要的代码和数据通过一个对象作为构造函数 实参传递进去，这个对象就是一个实现了Runnable接口的类的实例。
   2. 参考文献
      1. <https://www.cnblogs.com/victory8023/p/5549284.html>