1. join()方法
2. 概述

当主线程创建并启动子线程时，如果子线程耗时比较长，导致主线程在子线程之前结束了。如果我们想让主线程在子线程之后结束，就需要使用join()方法了，它的作用是等待线程对象销毁。如在主线程a中调用子线程b的start()方法，同时调用了b的join()方法，那么线程a就会进入无限期的阻塞中，等待线程b执行完毕并销毁后才能继续执行线程a。

1. join()与interrupt()

在当前线程中调用join()方法的过程中，如果当前线程对象被中断，则当前线程会出现异常，注意，是当前线程哦，不是调用join()方法的那个线程。即join()与interrupt()方法碰到，会出现异常。

1. join(long)与Thread.sleep(long)

设定等待时间的join()方法，内部使用wait(long)方法实现的，所以使用join(long)方法具有释放锁的特点，只要执行了join(long)方法，就会释放当前线程的锁，那么其他线程就可以调用此线程的同步方法了。

而Thread.sleep(long)方法是不释放锁的，它会一直持有该线程的锁，直到该线程代码执行结束才会释放该线程锁。

1. join(long)的执行时机

join(long)方法在线程中通常是先运行的，也就是先抢到该线程的锁，但是会快速地释放掉，来阻塞join(long)方法后该线程的代码执行。之后，join(long)会和其他线程来争抢该线程锁，谁抢到，谁就先执行。

1. ThreadLocal类
2. 概述

ThreadLocal类用于每个线程私有变量值的共享，它可以让每个线程绑定自己的数据，隔离不同线程的数据。

1. set()和get()

从函数的语义就可以知道，一个是往ThreadLocal类中写入值，一个是取出值。在不同的线程中调用这些方法，存入和取出的都是该线程的私有数据。

1. get()的默认值

初次调用get()方法时，如果没有值会返回null，我们可以为ThreadLocal类设置默认值，解决这个问题。通过继承ThreadLocal类，重写initialValue()方法，返回默认值。

public class ThreadLocalText extends ThreadLocal{

@Override

protected Object initialValue(){

return “默认值”;

}

}

1. InheritableThreadLocal类

通过类InheritableThreadLocal可以在子线程中取得父线程继承下来的值。如创建一个类local来继承InheritableThreadLocal类，在Main主线程中创建local类实例，然后存入一个值，创建一个新的线程，并从该线程中取出local类实例中的值，发现这两个值是一致的，说明了父子的继承关系。

1. 子线程修改父线程的值

通过继承InheritableThreadLocal类，还可以修改父线程存入的值，重写childValue()方法即可，传入父线程的值，返回修改后的值。

@Override

protected Object childValue(Object parentValue){

return parentValue +”123”;

}

注意，如果子线程在取得值并进行修改的同时，父线程将该实例中的值进行了修改，那么子线程获取到的依然是旧值。