1. 线程生命周期有哪些？

答：线程的生命周期包括新建状态（New）、可运行状态（Runnable）、阻塞状态（Blocked）、等待状态（Waiting）、超时等待状态（Time Waiting），死亡状态（dead）

1. 在多个线程访问同一个资源时，可以使用（ synchronized）关键字来实现线程同步，保证对资源安全访问
2. 创建使用ThreadPoolExecutor创建一个线程池，要求corePoolSize=10 maximumPoolSize=20 keepAliveTime=20 时间单位为秒 任务队列长度为200

import java.util.concurrent.\*;

public class ThreadPoolExecutorCreator {

public static void main(String[] args) {

ThreadPoolExecutor threadPoolExecutor = new ThreadPoolExecutor(10, 20, 20, TimeUnit.SECONDS,

new LinkedBlockingDeque<>(200),

new ThreadPoolExecutor.AbortPolicy());

final CountDownLatch countDownLatch = new CountDownLatch(4);

for (int i = 0; i < 4; i++) {

final int currentIndex = i + 1;

System.out.println("提交第" + currentIndex+ "个线程");

threadPoolExecutor.execute(() -> {

System.out.println(Thread.currentThread().getName()

+ ", currentIndex is:"

+ currentIndex);

countDownLatch.countDown();

});

}

System.out.println("提交完毕");

try {

countDownLatch.await();

} catch (InterruptedException e) {

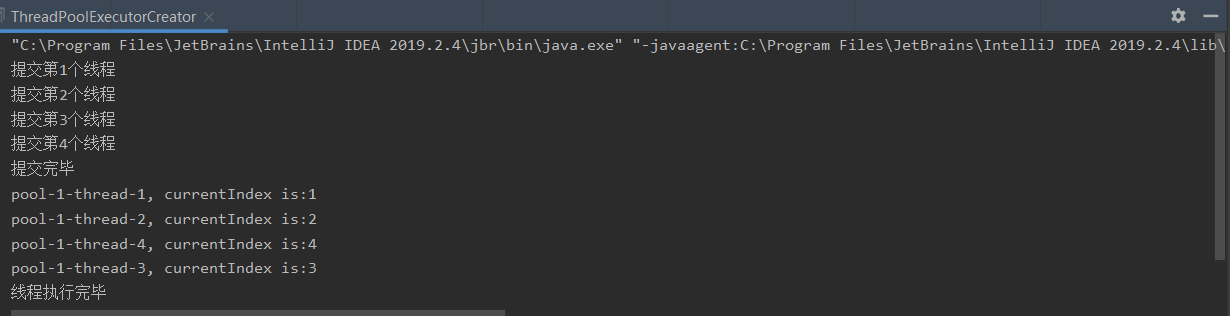
e.printStackTrace();

}

System.out.println("线程执行完毕");

}

}



1. 将课件中三种线程的实现方式 比着手敲一下

继承Thread类：

public class ThreadTest extends Thread{

@Override

public void run() {

synchronized (this) {

System.out.println(Thread.currentThread().getName());

}

}

public static void main(String[] args) {

ThreadTest t1 = new ThreadTest();

ThreadTest t2 = new ThreadTest();

ThreadTest t3 = new ThreadTest();

ThreadTest t4 = new ThreadTest();

t1.start();

t2.start();

t3.start();

t4.start();

}

}



实现Runnable接口：

public class RunnableTest implements Runnable {

@Override

public void run() {

synchronized (this){

System.out.println(Thread.currentThread().getName());

}

}

public static void main(String[] args) {

RunnableTest runnableTest = new RunnableTest();

new Thread(runnableTest, "线程1").start();

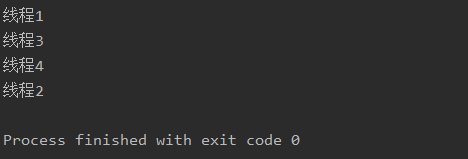
new Thread(runnableTest, "线程2").start();

new Thread(runnableTest, "线程3").start();

new Thread(runnableTest, "线程4").start();

}

}



实现Callable接口

import java.util.concurrent.Callable;

import java.util.concurrent.FutureTask;

public class CallableTest implements Callable<Integer> {

@Override

public Integer call() throws Exception {

synchronized (this){

System.out.println(Thread.currentThread().getName());

}

return null;

}

public static void main(String[] args) {

CallableTest callableTest = new CallableTest();

FutureTask futureTask1 = new FutureTask(callableTest);

FutureTask futureTask2 = new FutureTask(callableTest);

FutureTask futureTask3 = new FutureTask(callableTest);

FutureTask futureTask4 = new FutureTask(callableTest);

Thread thread1 = new Thread(futureTask1);

Thread thread2 = new Thread(futureTask2);

Thread thread3 = new Thread(futureTask3);

Thread thread4 = new Thread(futureTask4);

thread1.start();

thread2.start();

thread3.start();

thread4.start();

}

}



1. 定义了一个Student类，该类有一个属性id，为学生的编号。

定义了一个TakeTemperatureThread类，该类有一个方法takeTemperature，为学生测量体温。该类继承了Thread类并重写了run方法，实现了多线程。

在该类的main方法中，创建了一个学生列表，并启动5个线程实现为100个学生测体温的功能。