**集美大学计算机工程学院实验报告**

|  |  |  |
| --- | --- | --- |
| **课程名称**：Java程序设计 | **班级**：软件1912 | **实验成绩**： |
| **指导教师**：李　旺 | **姓名**：游嘉升 |  |
| **实验项目名称**：多线程 | **学号**：201921122050 | **上机实践日期**：12月9日 |
| **实验项目编号**：实验9 | **组号**： | **上机实践时间**： 2 学时 |

**一、目的**

掌握线程的创建与启动；了解线程同步与互斥。

**二、实验内容与设计思想**

1. 线程的创建与启动

邮件客户端(如foxmail)支持多帐号多线程，可以在接收邮件的同时进行邮件发送。不考虑多帐号的问题，编写ReceiveMailRunnable与SendMailRunnable类，均实现Runnable接口。这两个类主要有如下功能：

(1)分别打印"I am receiving emails" 和"I am sending emails"；

(2)休眠1000ms-2000ms间的随机时间(Thread.sleep)；

(3)循环执行上述(1)、(2)内容，循环10-20次(使用随机数实现)；

(4)退出时打印"当前线程XXX正要退出"，XXX为当前线程的名称(Thread.currentThread.getName())。

编写TestThread类，在main方法中启动3个ReceiveMailRunnable与3个SendMailRunnable线程。希望当这3个ReceiveMailRunnable与3个SendMailRunnable线程结束，才执行最后一句代码，打印"foxmail任务结束"(使用join)。

2. 并发计算模拟

计算从1到1亿整型数相加。要求使用并发程序处理，即采用多线程实现，在主线程中将计算结果累加(不能使用累加公式)。

大致思路：

(1)编写SumWorker类，实现Runnable接口，计算从m到n的和，其中m,n由构造方法传入；

(2)在主程序中每次开启若干个SumWorker线程(具体数量由程序指定或者用户输入)，计算完成之后，将部分结果累加，然后再启动另一批线程，直到计算完成。

说明：本题有点类似“网格计算”或者“云计算”。

扩展（选做）：学有余力的同学可以尝试研究参数对并行性能的影响，如：同时并发线程数、每线程计算数量、共享式sum与独立式sum等，并尝试分析其原因。

3. 哲学家就餐问题

哲学家就餐问题是1965年由Dijkstra提出的一种线程同步的问题。

一圆桌前坐着5位哲学家，两个人中间有一只筷子，桌子中央有面条。哲学家思考问题，当饿了的时候拿起左右两只筷子吃饭，必须拿到两只筷子才能吃饭。上述问题会产生死锁的情况，当5个哲学家都拿起自己右手边的筷子，准备拿左手边的筷子时产生死锁现象。

请尝试使用多线程模拟，确保不会出现“饿死”状态。

**三、实验使用环境**

Java version "13.0.2"

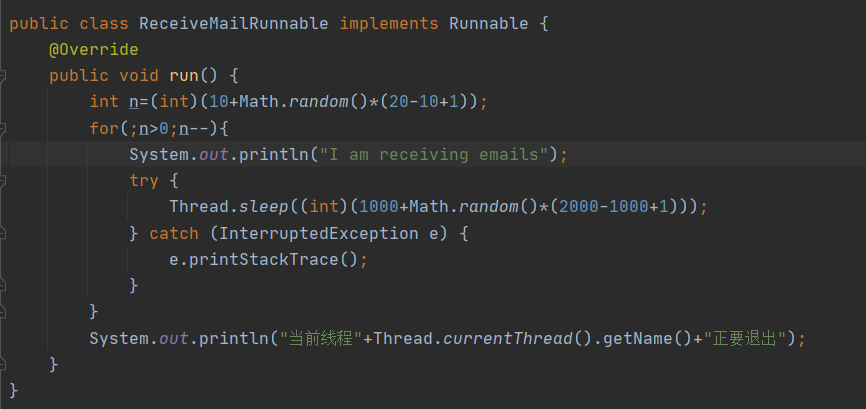
IntelliJ IDEA 2020.2.2 x64

**四、实验步骤和调试过程**

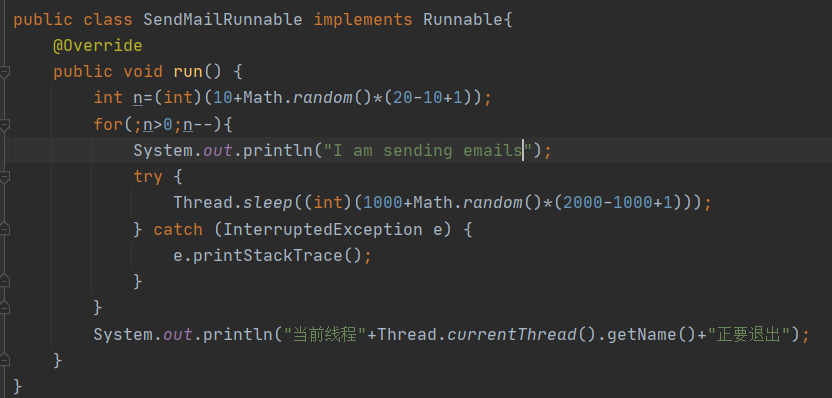
1. 线程的创建与启动

1. 设计思想：创建ReceiveMailRunnable与SendMailRunnable类实现Runnable接口，在run方法中实现间隔随机时间打印接收和发送信息。在Test类中创建六个线程实现题目测试，最后用join方法等待所有线程结束后打印结束信息。
2. 关键代码：

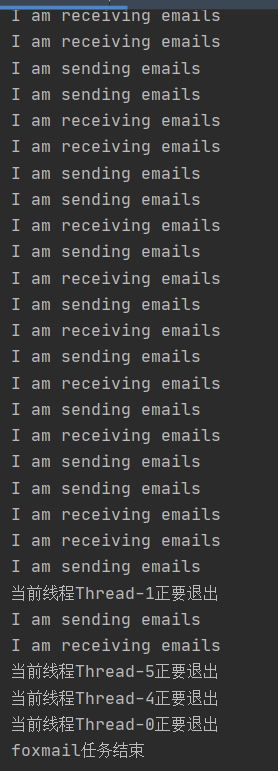
ReceiveMailRunnable类的实现



SendMailRunnable类的实现



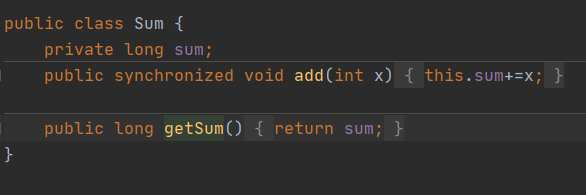
1. 测试数据：无测试数据
2. 测试结果：



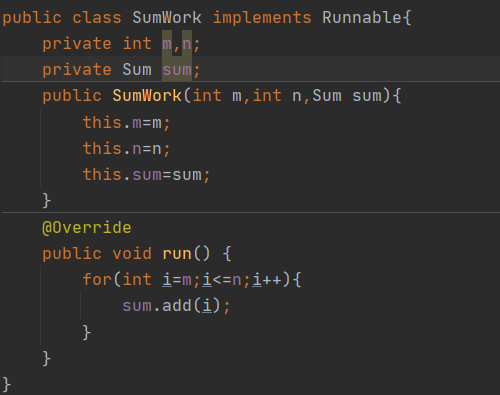
2. 并发计算模拟

1. 设计思想：创建Sum类并实现synchronized的add方法，使每次进入方法时方法有锁，创建SumWorl类实现run方法调用主方法创建的Sum对象的add方法，实现将区间的每一个数相加。
2. 关键代码：

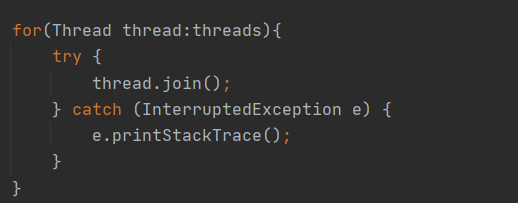
Sum类的实现



SumWork类的实现



等待各线程计算完毕



1. 测试数据：

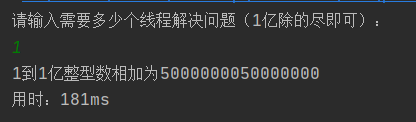
1

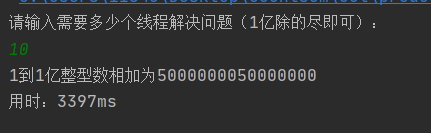
10

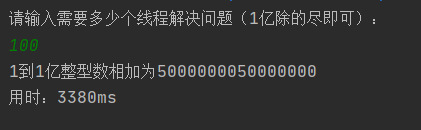
100

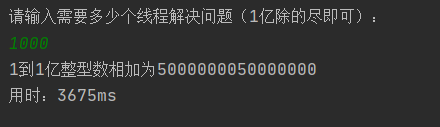
1000

1. 测试结果：





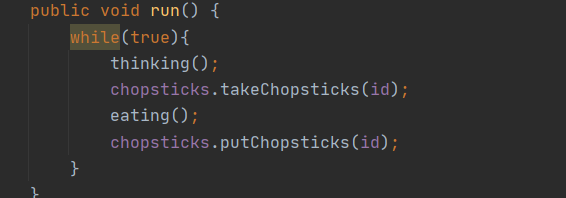




3. 哲学家就餐问题

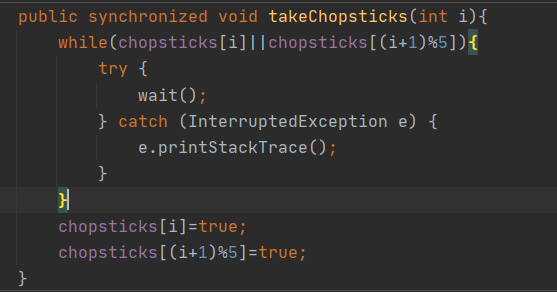
1. 设计思想：定义规则当哲学家左右手筷子都空闲时，哲学家才可以进餐，当哲学家进餐后必须同时放下左右手筷子，遵循这个规则设计就不会产生死锁。
2. 关键代码：

Philosopher类继承Runnable的run方法

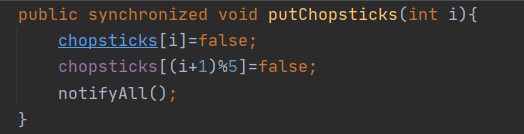


Chopsticks类实现的takeChopsticks方法

（在左筷子或者右筷子被占用时进入等待）



Chopsticks类实现的putChopsticks方法



1. 测试数据：无
2. 测试结果：



**五、实验小结**

1．实验中遇到的问题及解决过程

2．实验中产生的错误及原因分析

①第二题sum的输出结果与预期结果不符

解决过程：调试后后发现未使用join方法等待各线程结束运行再输出结果导致还未将所有结果加上得到错误输出。

②第二题中理论上线程越多，计算的速度会越快，而实际与之不符有点奇怪

解决过程：尚未解决。

3．实验体会和收获。

多线程的使用可以很大程度上使程序更加灵活，让编程逻辑更符合人类直觉，但是多线程的出现也导致了更多的线程安全问题，所以在使用多线程的时候要合理规避以及解决线程安全问题。

**六、附录**

[1][美] 凯.S.霍斯特曼（Cay S. Horstmann）.Java核心技术·卷 I（原书第10版）[M].机械工业出版社