**深 圳 大 学 实 验 报 告**

**课程名称：Java程序设计**

**实验项目名称： 必实验4 线程应用**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师：潘微科**

**报告人： 刘俊楠 学号：2017303010 班级： 01**

**实验时间： 2021年11月19日（周五）~2021年12月1日（周三）**

**实验报告提交时间： 2021.11.26**

**教务部制**

|  |
| --- |
| **实验目的与要求：**  **实验目的：** 掌握Java程序设计中的线程同步等技术。  **实验要求：**  (1). 运行以下三个程序（每个程序运行10次），并对输出结果给出分析。在报告中附上程序截图和详细的文字说明。（15分）**（程序有点特别，与预想不同，请分析为什么有这个结果）**  **程序1:**  _@[D~MWFEFC]P(KX1$ZP~EH  {2L`7D7X5LRWB81%NDX%4HF  ~[C63JZW%{XUJHK7Y@1L~R2  **程序2:**  _@[D~MWFEFC]P(KX1$ZP~EH  {2L`7D7X5LRWB81%NDX%4HF  D:\Program Files\WeChat Files\AppData\Roaming\Tencent\Users\1926768129\QQ\WinTemp\RichOle\]LKDQS~54D1LNKJ[]V`O92G.png  **程序3:**    (2). 编写Java应用程序实现如下功能：**第一个线程**输出数字1,2,..,12，**第二个线程**输出英文单词数字和月份One January, Two February, …, Twelve December，输出的顺序和格式为1OneJanuary2TwoFebruary...12TwelveDecember，即每1个数字紧跟着2个英文单词的方式。要求线程间实现通信。要求采用实现Runnable接口和Thread类的构造方法的方式创建线程，而不是通过Thread类的子类的方式。在报告中附上程序截图、运行结果截图和详细的文字说明。（15分）**（两线程需合作，顺序严格遵守，runnable接口）**  (3). 编写Java应用程序实现如下功能：创建工作线程，模拟银行现金账户取款操作。多个线程同时执行取款操作时，**如果不使用同步处理，会造成账户余额混乱，要求使用syncrhonized关键字同步**代码块，以保证多个线程同时执行取款操作时，银行现金账户取款的有效和一致。**要求采用实现Runnable接口和Thread类的构造方法的方式**创建线程，而不是通过Thread类的子类的方式。在报告中附上程序截图、运行结果截图和详细的文字说明。（25分）  (4). 有一座东西向的桥，只能容纳一个人，桥的东边有20个人（记为E1,E2,…,E20）和桥的西边有20个人（记为W1,W2,…,W20），编写Java应用程序**让这些人到达对岸**，**每个人用一个线程表示**，桥为**共享资源**，在过桥的过程中输出谁**正在**过桥**（不同人之间用逗号隔开）**。运行**10**次，**分别统计东边和西边的20人先到达对岸的次数**。要求采用实现Runnable接口和Thread类的构造方法的方式创建线程，而不是通过Thread类的子类的方式。在报告中附上程序截图、运行结果截图和详细的文字说明。（25分）  报告写作。要求：主要思路有明确的说明，重点代码有详细的注释，行文逻辑清晰可读性强，报告整体写作较为专业。（20分）  **说明：**  （1）本次实验课作业满分为100分，占总成绩的比例（待定）。  （2）本次实验课作业截至时间2021年12月1日（周三）21:59。  （3）报告正文：请在指定位置填写，本次实验**需要单独提交源程序文件**（源程序单独打包在Blackboard中上传，不要包含外部导入的包）。  （4）个人信息：WORD文件名中的“姓名”、“学号”，请改为你的姓名和学号；实验报告的首页，请准确填写“学院”、“专业”、“报告人”、“学号”、“班级”、“实验报告提交时间”等信息。  （5）提交方式：截至时间前，请在Blackboard平台中提交。  （6）发现抄袭（包括复制&粘贴整句话、整张图），**抄袭者和被抄袭者的成绩记零分（含抄袭往届同学的作业）。**  （7）延迟提交，不得分；如有特殊情况，请于截至日期之后的48小时内发邮件到panweike@szu.edu.cn，并在邮件中注明课程名称、作业名称、姓名、学号等信息，以及特殊情况的说明，我收到后会及时回复。  （8）期末考试阶段补交无效。 |

|  |
| --- |
| (1). 运行以下三个程序（要求每个程序运行10次），并对输出结果给出分析。在报告中附上程序截图和详细的文字说明。（15分）  **程序1:**   * 详细文字说明 * 首先写了一个继承runnable接口的类PrintChar，私有char型 charToPrint是用于打印的字符， int times为要打印字符的次数，完成了接口的run方法，其主要用于打印字符times次。      * 与上一个类类似，不过它的lastNum是用来限制次数的，run方法写了输出从1到lastNum，中间有空格。      * 公有类里的Main方法主要创建了三个Runnable类，并且用上图两个类建立对象，并且创建了三个关于三个runnable对象的线程。      * 结果分析 * （第一次运行）从下图（部分）我们可以看到，此次结果是先输出100次a再输出100次b，再从1输出到100，说明此次三个线程运行顺序是:thread1、thread2、thread3.（在我运行的多次中，大部分结果都与此一致）      * 第二次输出：先thread2，再thread1，在thread3      * 第三次输出：先thread2在thread3再thread1      * 第四次输出：正常      * 第五次输出：thread2->thread3->thread1      * 第六次输出：thread2/3/1交替乱序输出      * 第七次输出：正常      * 第八次输出：thread1与thread2乱序输出，然后输出thread3      * 第九次输出：thread1/2/3乱序输出      * 第十次输出：thread1/2/3乱序输出      * 在以上9次的结果输出中，我们可以看到有时先运行了几十次thread2，输出了几十次b，然后接着运行了thread3，然后在中间的时候发生了thread1、thread3、thread2交替输出的异常结果，这说明这几个线程的运行先后顺序是不定的！这是因为线程也被赋予了时间片轮转算法，时间片轮转调度算法是一种最古老，最简单，最公平的且使用最广的算法。每个线程被分配一个时间段，称作它的时间片，即该线程允许进行的时间。这会导致所有线程因为各种因素的影响输出变得不可预测。所以我们需要线程的同步，防止这种错误的发生。   **程序2:**  1、详细文字说明   * 程序2的runnable类与程序1的类一致，于是在此暂且不表。      * 我们看到该程序的main函数，可以看到与程序1不同的是，main函数中使用了java.util.concurrent包中的ExecutorService，也就是java中的线程池，newFixedThreadPool方法创建了一个定长线程池，可控制线程最大并发数，在此设置成了3.使用execute方法分别创建了三个runnable类的对象并且作为线程异步执行。      * 结果分析 * 在运行时，发现结果并不如想象的一样按顺序执行，是由thread1、thread2 交替执行，然后thread3执行，最后thread2执行，完全乱序。      * 第二次发现结果依然是无序的。      * 第三次执行有正常情况，thread1先执行、其次是thread2，最后是thread3      * 第四次输出：正常      * 第五次输出：thread2->thread3->thread1      * 第六次输出：thread2/1交替乱序输出，然后输出thread3      * 第七次输出：正常      * 第八次输出：thread1与thread2乱序输出，然后输出thread3      * 第九次输出：thread1/2乱序输出，再输出thread3      * 第十次输出：正常      * 分析其产生这种输出情况不确定的原因，就是因为ExecutorService的newFixedThreadPool方法在创建定长线程池的时候参数选择了3，也就是线程池大小为3，如果线程超过3则需要等待，而我们的程序只有三个线程，所以会出现三个线程竞争的结果，如果设置为1的话（如下图），则不会出现竞争结果。        * 当可控制线程最大并发数设置为1以后，一定会根据execute先后来执行线程，这是因为execute是异步的，而且超过一个线程则会进入队列等待，不会互相竞争输出。   **程序3:**   * 详细文字说明 * 我们首先在public 主类中创建一个静态类Account，其主要作用是拥有一个balance变量，getBalance方法返回balance变量，deposit方法根据传入的参数更新balance，并且使线程睡眠5ms。      * 主类中创建了一个AddApennyTask类，其继承了runnable接口，并且重写了run方法，其内容为执行account的deposit方法，参数为1.      * 主类下也定义了一个Account 类的静态对象account，并且在main方法中使用了ExecutorService类的newCachedThreadPool方法，创建一个可缓存线程池，如果线程池长度超过处理需要，可灵活回收空闲线程，若无可回收，则新建线程。 * 接着执行100次for循环，每次创建一个新线程。循环结束后使用shutdown方法关闭所有进程，当未完全关闭时一直在while循环中循环，完全关闭后输出account的balance的值。     2、结果分析   * 我们可以看到，此处运行完，我们的balance只有10，而newCachedThreadPool方法创建的是可缓存线程池，那么答案应该是100，是什么地方出问题了呢？      * 第二次发现结果依然是10   C:\Users\11440\Documents\Tencent Files\1144097453\Image\SharePic\20211130204512.png   * 第三次执行更离谱，balance为5      * 第四次输出：结果为4      * 第五次输出：balance为5      * 第六次输出：balance为7      * 第七次输出：结果为4      * 第八次输出：结果为8      * 第九次输出：balance为7      * 第十次输出：balance为6      * 我们尝试单步调试走完这个程序，发现balance确实是100。      * 我们修改run方法，在run方法下输出当前线程的名称，并且输出account的balance值。      * 查看结果发现虽然是乱序运行的线程，但是都没有回收之前的线程，所以直接运行的话应该是运行完100个线程了的，但是balance不是立刻变化，而是在多个线程同一样的值，甚至在后面有减少的趋向！      * 我们可以看到，balance不会即时变化的原因在于run方法调用的deposit方法里面有线程休眠5ms，这会导致线程运行时间有先后顺序，再加上是可缓存线程池，就会导致后续线程所用的balance并不是真正上一个线程的balance，这导致了最后balance不增反减的结果。后续线程所用的balance还未即时更新，这导致了多个线程的balance为同一个值得结果。      * 当我们把deposit方法的线程睡眠语句去掉，发现结果更新明显正确了许多，但是还是会有偏差，这是因为可缓存线程池中的多个线程竞争balance变量时，也有可能导致其使用的balance未得到及时更新的问题，所以，线程之间的同步显得非常重要且不可忽视。     (2). 编写Java应用程序实现如下功能：第一个线程输出数字1,2,..,12，第二个线程输出英文单词数字和月份One January, Two February, …, Twelve December，输出的顺序和格式为1OneJanuary2TwoFebruary...12TwelveDecember，即每1个数字紧跟着2个英文单词的方式。要求线程间实现通信。要求采用实现Runnable接口和Thread类的构造方法的方式创建线程，而不是通过Thread类的子类的方式。在报告中附上程序截图、运行结果截图和详细的文字说明。（15分）  1、详细文字说明   * 首先书写一个Month类，该类储存了要输出的值与递增的num，拥有有参构造方法，初始化了用于控制输出的flag，并且实现了两个synchronized的输出相应数字num与相应数字月份英文的month 的方法。在输出数字的printint方法中，我通过flag来判断是否应该输出num，如果不是输出num的线程，则让其wait，在执行完该步骤后，通过notify来唤醒输出month的线程，这其中，synchronized实现了每次运行该方法时，别的线程会陷入等待，而flag实现了阻止我不想要输出的线程的功能。      * 接下来的Num类继承了Runnable接口，并且存储了两个线程的名称与上述Month类，实现两个线程共享一个数据，拥有一个有参构造方法，其中的run方法通过不断轮询，来输出数字或英文，通过Month类的flag，来判断是运行哪个方法，定位到了具体的线程。      * 接下来的main方法主要初始化了两个线程的名称，分别为int 与 month，然后创建了一个类 Month类 month，用于共享的数据与方法，并且初始化了一个runnable 的Num类，用于创建接下来的两个线程integ与month，接着，设置两个线程的名字，然后启动线程即可。     2、完整代码     * 结果分析 * 结果与题目要求一致，这是因为我利用了synchronized与flag实现了线程之间通信，通过wait与notifyall方法预防了线程出错的情况。     (3). 编写Java应用程序实现如下功能：创建工作线程，模拟银行现金账户取款操作。多个线程同时执行取款操作时，如果不使用同步处理，会造成账户余额混乱，要求使用syncrhonized关键字同步代码块，以保证多个线程同时执行取款操作时，银行现金账户取款的有效和一致。要求采用实现Runnable接口和Thread类的构造方法的方式创建线程，而不是通过Thread类的子类的方式。在报告中附上程序截图、运行结果截图和详细的文字说明。（25分）  1、详细文字说明   * 首先我们书写一个Bank类，用于存储数据以及运行同步方法，其中保存了用户名s1，以及上个线程、上上个线程名，类拥有一个有参构造函数，输出初始余额。 * 并且还写了一个同步synchronized取钱方法deposit，首先对线程名称进行判定，如果为上一个或者上上个线程名称，则让其等待，这样保证了三个线程的轮流输出，接下来我设置了balance每次取钱只减少10的设定，在if语句中输出哪个线程取了钱，并且输出当前银行余额，然后通知所有线程，因为是三个线程，所以通知一个总是会通知同一个线程，所以只能使用notifyall的方法通知所有线程就绪，然后根据当前线程更新线程名变量。      * 书写一个quqian类，继承了Runnable接口，并且拥有Bank类变量，用于run方法中调用deposit取钱方法，在类中实现了run方法，并且用while循环限定执行次数，其中调用Bank变量的deposit方法，让每个线程执行。      * 在main方法中，我们定义了用户名为刘俊楠，三个线程名字分别为资本家、小偷、生活，并且初始化bank类，定义用户名以及余额数量，然后使用声明三个Thread对象，并且设置其名称，最后启动三个线程。     2、完整代码      3、结果展示   * 如下图，我们可以看到首先我们先输出当前余额，然后三个线程轮流取钱，最后没钱时自动结束的结果。     (4). 有一座东西向的桥，只能容纳一个人，桥的东边有20个人（记为E1,E2,…,E20）和桥的西边有20个人（记为W1,W2,…,W20），编写Java应用程序让这些人到达对岸，每个人用一个线程表示，桥为共享资源，在过桥的过程中输出谁正在过桥（不同人之间用逗号隔开）。运行10次，分别统计东边和西边的20人先到达对岸的次数。要求采用实现Runnable接口和Thread类的构造方法的方式创建线程，而不是通过Thread类的子类的方式。在报告中附上程序截图、运行结果截图和详细的文字说明。（25分）  1、详细文字说明   * 首先创建一个Bridge类，主要存储数据以及建立synchronized方法，使每个线程能够共享数据与方法，在此处我主要创建了记录东西边过去的人的变量X\_FLAG与D\_FLAG，int数组[]occur装载着每个线程的运行情况，若已运行则让其wait，防止其重复过桥，string类型的name数组为装载每个线程的名称，用于比对，并且本类拥有一个无参构造方法，初始化occur数组与string数组。      * 在synchronized的cross方法中，我首先对线程进行比对，遍历我的name数组，找到对应的occur的位置，然后如果occur数组相应位置为1则说明线程已经运行过，于是让其wait。      * 在过桥执行完后我们选择Thread.sleep方法睡眠1秒钟，这是为了模拟过桥时间，接下来是一个if语句，判断过桥人数是否单边超过20，如果没超过则对相应的边增加人数，并且当做其已经过了桥，到此该线程已经执行完毕。如果过桥人数单边大于等于20个，则回车换行，最后我们根据人数确定哪边胜出，并且退出。      * 下图是继承了Runnable接口的一个People类，其有个有参构造函数以及一个Bridge类的变量，然后实现的Run方法中我们在任意一方过桥人数没有超过20人的时候就一直循环Birdge类变量B的cross方法。      * 在main方法中，我们建立了两个Thread类的数组对象，分别是[]dong、[]xi，其分别装载着20个线程，分别代表着东西边的20个人，接下来的两个string数组named与namex分别是东西两边各20个人的名称，接着我们实现了一个桥的对象b，然后根据b实现了People类对象p，通过p在20次for循环里分别对东西两边的线程做初始化，并且启动线程。     2、完整代码        3、结果分析   * 第一次运行输出：东家胜利      * 第二次输出：与第一次结果一样      * 第三次输出：西家胜出      * 第四次输出：东家胜利      * 第五次输出：东家胜利      * 第六次输出：东家胜利      * 第七次输出：东家胜利      * 第八次输出：东家胜利      * 第九次输出：东家胜利      * 第十次输出：东家胜利      * 通过观察结果我们可以知道，大部分线程都跟我的代码结构一样，显示东边线程然后西边线程，这是因为我在一个for循环里只启动了东西边各一个线程，但是即便如此也会出现差错，本来应该东边全胜的，却让西边赢了1局，这是因为线程竞争之间有很多因素影响，再分析输出我们发现，在过桥时中间阶段没有按照数字顺序顺序输出，而是从12开始输出，我认为是因为我的同步方法中加入了sleep方法导致。   ++++++++++++++++++++++++++++++++++++++++++++++++++++++  **其他（例如感想、建议等等）。**   * 通过本次实验，我切实感受到了线程同步的重要性，以及线程通信的难度，我认为这也是java的难点以及魅力所在。 * 明白了java的通信与同步是不同的，同步也有很多种，包括同步方法、同步代码块以及使用局部变量实现同步。 * 明白了java如何通过synchronized方法+wait()方法+notify()方法实现线程之间的通信。 |

深圳大学学生实验报告用纸

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  2021年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。