**深 圳 大 学 实 验 报 告**

**课程名称： Java 程序设计**

**实验项目名称：课程实验3：常用集合类和线程**

**学院： 计算机与软件学院**

**专业： 人工智能卓越班**

**指导教师： 潘微科**

**报告人：邓瑞霖 学号：2024150040 班级： 01**

**实验时间：2025年10月27日（周一）-2025年11月19日（周三）**

**实验报告提交时间： 2025年11月8日星期六**

**教务部制**

|  |
| --- |
| **实验目的与要求：**  **实验目的：**掌握常用的集合类，能够较为熟练的查阅Java提供的常见的类，并进行程序设计，掌握Java程序设计中的线程同步等技术。  **实验要求：**  **Part 1（25分）**  (1.1) 编写Java应用程序，实现浮点数（float）稀疏矩阵的乘法运算，其中稀疏矩阵是指矩阵中的绝大部分元素的值为0。在命令行读入和输出矩阵中的元素的时候，采用坐标格式，即(行号,列号): 数值，例如“(第0行, 第3列): 数值0.89”表示为(0, 1): 0.89。要求用下面的一个5\*4的矩阵和一个4\*6的矩阵为例，计算它们的乘积，并将结果以同样的坐标格式输出。在报告中附上程序截图、完整的运行结果截图和简要文字说明。（10分）  5\*4的矩阵：  (0, 0): 1.25  (0, 2): 0.75  (1, 1): 2.83  (1, 3): 1.47  (2, 0): 3.12  (2, 2): 0.92  (3, 3): 2.56  (4, 1): 4.31  (4, 3): 1.08  4\*6的矩阵：  (0, 0): 2.17  (0, 3): 0.89  (1, 1): 1.34  (1, 4): 3.75  (2, 2): 0.68  (2, 5): 2.41  (3, 0): 4.12  (3, 2): 1.53  (3, 4): 0.97  (1.2) 有12个国家（美国、中华人民共和国、德国、日本、英国、印度、法国、意大利、加拿大、韩国、以色列、俄罗斯），其属性有name、GDP2024和Olympics2024，分别表示国家名称、世界银行公布的2024年的国内生产总值（单位：百万美元）和在2024年巴黎奥会上获得的奖牌数量。  编写一个Java应用程序，要求使用TreeSet。（i）按照Olympics2024从大到小排序输出这些国家的信息；（ii）按照GDP2024从大到小排序输出这些国家的信息。要求以上(i)和(ii)两小题中的每小题都通过以下两种方式实现：通过实现Comparator接口或通过实现Comparable接口。在报告中附上程序截图、完整的运行结果截图和详细的文字说明。（15分）  **Part 2（25分）**  (2.1) 将第8章讲义（JavaPD-Ch08）中的5个应用程序（Example8\_1，Example8\_2，Example8\_3，Example8\_4，Example8\_6）在Eclipse或其他IDE中运行，如运行结果不唯一，则需要运行多次并至少得到两个不同的结果。对重要语句加上注释。在报告中附上程序截图、运行结果截图和简要文字说明（对运行结果做出解释）。（5分）  (2.2). 运行以下三个程序（每个程序运行5次），并对输出结果给出分析。在报告中附上程序截图和简要的文字说明（包括对结果的分析）。（5分）  **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    **程序3:**    (2.3) 第8章讲义（JavaPD-Ch08）中的第5个应用程序（Example8\_5）存在线程间不同步的问题，请修改该程序，以解决不同步的问题。在报告中附上程序截图、运行结果截图和详细的文字说明（包括设计的思路和合理性分析）。（10分）  (2.4) 编写一个演示死锁的程序，该程序包含两个线程。在报告中附上程序截图、运行结果截图和详细的文字说明（包括设计的思路和相应的分析，特别是为何导致死锁，以及如何消除死锁）。（5分）  **Part 3（30分）**  (3.1) 编写Java应用程序实现如下功能：第一个线程不停地随机生成[0,1)之间的浮点数（float）并输出到屏幕，第二个线程将第一个线程输出的第1-5个浮点数的平均值输出到屏幕（紧跟在第一个线程输出的第5个浮点数之后）、将第一个线程输出的第6-10个[0,1)之间的浮点数的平均值输出到屏幕（紧跟在第一个线程输出的第10个浮点数之后）…。要求通过synchronized、wait()和notify()实现线程间同步和通信。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（包括设计的思路和合理性分析）。（10分）  (3.2) 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和存款操作。多个线程同时执行取款和存款操作时，如果不使用同步处理，会造成账户余额混乱，要求通过synchronized、wait()和notifyAll()实现线程间同步和通信，以保证多个线程同时执行取款和存款操作时，银行现金账户取款和存款的有效和一致。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（假设银行存款有1000元，有2个取款线程和3个存款线程，每次取款和存款均为100元）、运行结果截图（显示每次存取款操作后的余额等信息，以说明线程间同步正确）和详细的文字说明。（10分）  (3.3) 有一座南北向的桥，只能容纳一个人，桥的南边有1000个人（记为S1,S2,…,S1000）和桥的北边有1000个人（记为N1,N2,…,N1000），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，分别统计南边的1000人和北边的1000人先全部到达对岸的次数（第i行输出格式为：第i次运行，南边/北边先完成过桥）。要求通过ReentrantLock()实现线程间同步和通信。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（包括对结果的分析）。（10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7%。  （2）本次实验课作业截至时间2025年11月19日（周三）21:59。  （3）报告正文：请在指定位置填写，本次实验不需要单独提交源程序文件。  （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。**  （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。 |

|  |
| --- |
| **Part 1（25分）**  (1.1) 编写Java应用程序，实现浮点数（float）稀疏矩阵的乘法运算，其中稀疏矩阵是指矩阵中的绝大部分元素的值为0。在命令行读入和输出矩阵中的元素的时候，采用坐标格式，即(行号,列号): 数值，例如“(第0行, 第3列): 数值0.89”表示为(0, 1): 0.89。要求用下面的一个5\*4的矩阵和一个4\*6的矩阵为例，计算它们的乘积，并将结果以同样的坐标格式输出。在报告中附上程序截图、完整的运行结果截图和简要文字说明。（10分）  5\*4的矩阵：  0  4\*6的矩阵：  (0, 0): 2.17  (0, 3): 0.89  (1, 1): 1.34  (1, 4): 3.75  (2, 2): 0.68  (2, 5): 2.41  (3, 0): 4.12  (3, 2): 1.53  (3, 4): 0.97  首先初始化两个矩阵：      将上述元素添加到矩阵里面；  矩阵1:matrix1    矩阵2:matrix2：    进行矩阵乘法：    输出结果：（结果保留两位数    (1.2) 有12个国家（美国、中华人民共和国、德国、日本、英国、印度、法国、意大利、加拿大、韩国、以色列、俄罗斯），其属性有name、GDP2024和Olympics2024，分别表示国家名称、世界银行公布的2024年的国内生产总值（单位：百万美元）和在2024年巴黎奥会上获得的奖牌数量。  编写一个Java应用程序，要求使用TreeSet。（i）按照Olympics2024从大到小排序输出这些国家的信息；（ii）按照GDP2024从大到小排序输出这些国家的信息。要求以上(i)和(ii)两小题中的每小题都通过以下两种方式实现：通过实现Comparator接口或通过实现Comparable接口。在报告中附上程序截图、完整的运行结果截图和详细的文字说明。（15分）  实现Country类：成员变量：name, GDP2024, Olympics2024    成员变量的get，set方法：    i）1）通过Comparable实现：  Country类实现接口Comparable    重写public int compareTo(Object o);方法；    主方法实现：    2）通过Comparator实现  实现比较类：    主函数：    输出均为：    ii）1）通过Conparable实现  Country类的public int compareTo(Objcet o)方法：    main方法：     1. 通过Comparator实现     main方法：    输出均为：    **Part 2（25分）**  (2.1) 将第8章讲义（JavaPD-Ch08）中的5个应用程序（Example8\_1，Example8\_2，Example8\_3，Example8\_4，Example8\_6）在Eclipse或其他IDE中运行，如运行结果不唯一，则需要运行多次并至少得到两个不同的结果。对重要语句加上注释。在报告中附上程序截图、运行结果截图和简要文字说明（对运行结果做出解释）。（5分）  **Example8\_1**  程序代码：    第一次运行结果：    第二次运行结果：    两次的运行结果不同，但相同的是都是先运行Main Thread  原因：因为线程的执行顺序是由操作系统（线程调度器）决定的，具有不确定性，程序员无法精确控制。  **Example8\_2**  程序代码：    第一次运行：    第二次运行：    两次的运行结果不同  **Example8\_3**  程序代码：      第一次运行：    第二次运行：    两次运行结果不同，但都没办法终止运行  **Example8\_4**  程序代码：      运行结果：（每次运行结果均相同）    **Example8\_6**  程序代码：      运行结果：（每次运行结果都相同）    (2.2). 运行以下三个程序（每个程序运行5次），并对输出结果给出分析。在报告中附上程序截图和简要的文字说明（包括对结果的分析）。（5分）  **程序1:**  程序代码：      运行第一次： 运行第二次： 运行第三次：    运行第四次： 第五次运行    结果分析：  从输出结果上看：  ·字母’a’, ‘b’ 和数字交替出现，没有固定的先后顺序  ·同一线程的多次输出被其他线程的输出打断  ·每次运行的线程执行顺序都不相同  ·线程切换点随机，没有固定模式  ·CPU时间分配具有不确定性  **程序2:**  程序代码：      运行第一次： 运行第二次： 运行第三次：    运行第四次： 运行第五次：    对运行结果进行分析：  从输出结果可以看出典型的并发执行模式：  ​输出交错混合：字母'a'、'b'和数字完全混合在一起，没有固定的顺序  ​线程交替执行：例如序列"a → b → 1 → a → 2 → b"展示了三个线程在快速切换  ​非确定性调度：每次运行的输出序列都不同，证明线程调度具有随机性  **程序3:**  程序代码：  两个内置类：Account和AddAPennyTask      main方法：    输出结果：  五次输出均为：    对运行结果进行分析：  ​严重的数据一致性破坏：99次存款操作完全丢失  ​典型的竞态条件：多个线程对共享资源的非同步访问  (2.3) 第8章讲义（JavaPD-Ch08）中的第5个应用程序（Example8\_5）存在线程间不同步的问题，请修改该程序，以解决不同步的问题。在报告中附上程序截图、运行结果截图和详细的文字说明（包括设计的思路和合理性分析）。（10分）  不同步的原因：   1. 原程序多个线程同时修改number，产生竞态条件 2. 线程未同步   设计思路：   1. 在Task类中设计两个私有成员变量 2. 使用synchronized修饰一个方法，并在该方法内在特定时机调用wait()和notifyAll()   程序代码（修改后的）：  Task类：设计两个私有成员变量number1, number2    使用synchronized修饰一个方法calcNumber()用来同步线程，并在特定的时机调用wait()方法，在程序结束时调用notifyAll()通知所有处于wait()的线程：    main方法:    输出结果：    由结果可看出，该程序解决了线程同步的问题，即正负数依次输出  (2.4) 编写一个演示死锁的程序，该程序包含两个线程。在报告中附上程序截图、运行结果截图和详细的文字说明（包括设计的思路和相应的分析，特别是为何导致死锁，以及如何消除死锁）。（5分）  程序代码：      运行结果：    程序设计思路：  共享资源：定义两个共享对象 resourceA 和 resourceB，代表两把锁。  线程1：先获取 resourceA 再尝试获取 resourceB。  线程2：先获取 resourceB 再尝试获取 resourceA。  产生死锁条件：  当线程1持有A等待B，而线程2持有B等待A时，双方都无法继续运行，程序卡死。  死锁的产生满足以下四个必要条件：  互斥条件：资源一次只能被一个线程占用；  请求与保持条件：线程已持有一个资源，同时又请求另一个资源；  不剥夺条件：资源不能被强行剥夺；  循环等待条件：线程之间形成资源等待的环路（T1 等 T2 的资源，T2 等 T1 的资源）。  在本程序中，这四个条件全部成立，所以出现了死锁。  解决死锁的思路：  要避免死锁，只需破坏其中任意一个条件  例如：让所有线程都按照相同的顺序申请锁资源，例如都先锁A再锁B。    这样不会产生循环等待  或使用 ReentrantLock.tryLock() 等超时机制避免无限等待。  **Part 3（30分）**  (3.1) 编写Java应用程序实现如下功能：第一个线程不停地随机生成[0,1)之间的浮点数（float）并输出到屏幕，第二个线程将第一个线程输出的第1-5个浮点数的平均值输出到屏幕（紧跟在第一个线程输出的第5个浮点数之后）、将第一个线程输出的第6-10个[0,1)之间的浮点数的平均值输出到屏幕（紧跟在第一个线程输出的第10个浮点数之后）…。要求通过synchronized、wait()和notify()实现线程间同步和通信。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（包括设计的思路和合理性分析）。（10分）  程序代码：  创建一个Task类调用Runnable接口：  定义私有成员变量name1，name2分别表示两个线程，并命名为first, second  定义一个列表用来存放线程一生成的浮点数  定义计数值count来判断线程一生成了多少浮点数    run方法：  使用synchronized来同步线程，线程一当count==5时，使用wait()中断方法执行，线程2若count!=5，即线程一产生的浮点数个数<5，使用wait()中断方法执行      main方法：    输出结果：  部分结果如图所示：    (3.2) 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和存款操作。多个线程同时执行取款和存款操作时，如果不使用同步处理，会造成账户余额混乱，要求通过synchronized、wait()和notifyAll()实现线程间同步和通信，以保证多个线程同时执行取款和存款操作时，银行现金账户取款和存款的有效和一致。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（假设银行存款有1000元，有2个取款线程和3个存款线程，每次取款和存款均为100元）、运行结果截图（显示每次存取款操作后的余额等信息，以说明线程间同步正确）和详细的文字说明。（10分）  程序代码：  创建一个类：TaskBank并调用Runnable接口  成员变量：money初始化为1000    run方法，用synchronized同步线程：    main方法：    运行结果；    (3.3) 有一座南北向的桥，只能容纳一个人，桥的南边有1000个人（记为S1,S2,…,S1000）和桥的北边有1000个人（记为N1,N2,…,N1000），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，分别统计南边的1000人和北边的1000人先全部到达对岸的次数（第i行输出格式为：第i次运行，南边/北边先完成过桥）。要求通过ReentrantLock()实现线程间同步和通信。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（包括对结果的分析）。（10分）  程序代码：            运行结果（部分）：    实验结果分析：  在程序运行 10 次的实验中，南北两侧各有 1000 个线程同时尝试过桥。  由于桥被 ReentrantLock 控制为同一时刻仅允许一个线程通行，所以输出中可以看到过桥的过程是严格串行的：每次只有一个人显示“正在过桥”。  ++++++++++++++++++++++++++++++++++++++++++++++++++++++  **其他（例如感想、建议等等）。**  通过本次课程实验，我更加深入地理解了多线程编程、线程同步机制以及共享资源互斥访问的重要性。  在编写桥梁过桥模拟程序时，我体会到线程调度的随机性与不可预测性，也认识到合理使用 ReentrantLock、synchronized、wait()、notify() 等机制能够有效避免线程竞争带来的混乱与死锁问题。  整个实验让我从理论走向实践，加深了对 Java 并发模型 的理解。  尤其是在调试过程中，我学会了如何分析程序执行的顺序问题，如何通过 AtomicInteger 等原子类保证数据一致性，这对我今后编写并发程序有很大帮助。  此外，通过阅读并运行教材中的示例程序，我也体会到线程优先级、锁的公平性、以及线程间通信机制的差异，进一步增强了自己独立思考和解决问题的能力。 |

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：  2025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。