**深 圳 大 学 实 验 报 告**

**课程名称： Java 程序设计**

**实验项目名称： 实验3 常用集合类和线程**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 姚俊梅**

**报告人： 杜良衡 学号： 2022150255 班级： 01B**

**实验时间：2024年10月29日 ~ 2024年11月23日**

**实验报告提交时间： 2024.11.05**

**教务部制**

|  |
| --- |
| **实验目的与要求：**  **实验目的：**掌握Java程序设计开发环境的搭建，掌握编译、运行等基本步骤和命令；掌握基础知识和基本类型应用，以及类的初级应用。  **实验要求：**  1、String类的应用以及正则表达式的使用（20分，每题10分）  (1) 编写一个Java应用程序，读取一个以空格分隔的字符串，并将其解析为一个包含单词数量的统计信息。每个单词的统计信息包括单词本身和该单词出现的次数。具体要求如下：编写一个方法 countWords(String input)，对输入input字符串进行解析并返回另外一个格式为"word1: count1; word2: count2; ...; wordN: countN"的字符串。请用如下字符串进行测试：  "Tom and Jerry is an American animated media franchise and series of comedy short films created in 1940 by William Hanna and Joseph Barbera. Best known for its 161 theatrical short films by Metro-Goldwyn-Mayer, the series centers on the enmity between the titular characters of a cat named Tom and a mouse named Jerry. Many shorts also feature several recurring characters."  在WorldCounter类中先创建方法countWords用来解析字符串为一个包含单词数量的统计信息，参数input为待解析的字符串。该方法内先调用String.split方法以空格为分隔符分割字符串，并初始化HashMap对象用来存储单词出现次数。然后遍历分割后的字符串数组words。对每个单词，若wordCount中存在，则出现次数加一，否则新增键值对到workCount中，key为该单词本身，value为默认值1。  统计完成后需要构建格式化输出的字符串并返回。为方便字符串操作选用StringBuilder来构建对象result。接着遍历wordCount中每个键值对，取出key和value值并调用append方法来附加到result序列，需注意要去掉最后的“; ”，直接将字符串长度减2即可。最后返回统计得到的字符串。  在该类的main方法中，先创建了待解析的字符串testInput，再调用countWords进行解析并输出统计信息。  运行该程序，可以看到程序成功统计了输入字符串中每个单词的出现次数。  (2) 编写一个Java应用程序，每次从键盘读入一个包含有大写英文字母、小写英文字母和数字混杂的字符串(例如Aa123bEFGaaa49023)，要求按反转顺序输出大写英文字母（例如GFEA）、小写英文字母（aaaba）和数字（32094321）。要求循环连续测试10次，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。  先编写ReverseCharacters类的主方法，该方法循环十次从键盘读取字符串，并调用processInput方法来处理并输出结果字符串。  processInput方法用于处理输入的字符串，分离并反转输出大小写英文字母和数字。该方法内先分别创建3个StringBuilder对象来存储大小写字母和数组。然后为实现反转输出，从后往前遍历输入字符串。对遍历到的每一个字符，判断它属于大写字母、小写字母还是数字，并附加到对应的序列。  处理完成后输出这三个结果字符串。  运行程序，循环连续测试10次，结果符合预期。    2、集合类的应用（40分，每题10分）  (1) 编写一个Java应用程序，读取一个CSV格式的字符串并将其解析为一个包含学生信息的对象列表。每个学生的信息包括姓名、年龄和成绩。要求如下：i) 首先定义一个 Student 类，包含name（String，表示学生的姓名）、age（int，表示学生的年龄）及score（float，表示学生的成绩）；ii) 编写一个 StudentParser 类，包含方法List<Student> parseStudents(String csvData)，该方法接收一个以逗号分隔的字符串（CSV格式），并返回一个 Student 对象的列表。请使用以下CSV格式的字符串进行测试：  Alice,20,88.5  Bob,22,75.0  Charlie,19,90.0  先定义Student类，其中包含name、age和score三个成员变量，用于初始化成员变量的有参构造方法，以及重写的toString方法来格式化输出学生信息。  接着编写StudentParser类，先创建parserStudents方法用来读取并解析CSV格式字符串为包含学生信息的对象列表。  在该方法内，先初始化学生对象列表。因为CSV格式文件的每一行都是一个数据记录，每个记录由一个或多个字段组成，用逗号分隔，所以先以换行符为分隔符分割字符串每一行，然后以逗号为换行符分割每一行数据，确保每一行包含学生姓名、年龄和成绩三个字段。通过分割得到的字段构建Student对象并添加到列表中。最后返回解析得到的列表。  在StudentParser类的main方法中进行测试。编写CSV格式字符串，创建StudentParser实例来调用parseStudents方法解析字符串，最终输出得到的所有学生的信息列表。  运行程序，成功解析并打印出学生信息列表。  (2) 编写Java应用程序，使用集合类管理图书信息。该系统能够存储图书信息、查找特定图书以及列出所有图书。要求如下：  i) 创建Book类，包含属性title（String，图书标题）、author（String，作者姓名）及isbn（String，ISBN号）。  ii) 创建BookManager类，使用 HashSet<Book> 来存储图书信息，提供的方法包括void addBook(Book book)用于将图书添加到集合中，Book findBookByTitle(String title) 根据标题查找并返回对应的图书，方法 void listAllBooks() 输出所有存储的图书信息。  iii) 在主类的 main 方法中创建 BookManager 实例，并演示添加、查找和列出图书的功能。请使用如下两种图书信息进行测试：  《Java编程思想》，作者：Bruce Eckel，ISBN：978-7-111-23866-3  《Effective Java》，作者：Joshua Bloch，ISBN：978-7-121-29968-0  先创建Book类，其中包含title、author和isbn三个成员变量，用于初始化成员变量的有参构造器，用来返回私有成员变量title值的getter方法，以及重写的toString方法，用来格式化输出图书信息。  接着创建BookManager类，先定义HashSet<Book>类型的变量books来存储图书信息。接着构建添加图书到集合的addBook方法，其会调用Hashset的add方法来进行添加，该方法会保证集合内不会有重复元素。  再创建findBookByTitle方法来根据标题查找对应的图书，该方法会遍历books集合，并调用Stirng的equal方法来比较标题是否相等，因为String作为引用类型的变量，应比较的是对象的内容是否相等，而如果使用“==”操作符比较的是两个对象的地址。  最后创建listAllBooks来输出所有存储的图书信息。  在主类Test的main方法中进行测试。先创建BookManager实例manager用来存储图书信息，再创建两个Book对象并调用addBook方法添加到对象manager中。  接着查找指定标题的图书，若查询到则打印图书信息，若无则输出查询失败。最后调用listAllBooks列出所有图书。  运行程序，可以看到添加、查找和列出图书的功能均符合预期。  (3) 编写一个Java应用程序，按照GDP从小到大排序输出10个国家的信息，要求：i) 创建一个Country类，包括属性name、gdp2023和population，分别表示国家名称、2023年的国内生产总值（单位：百万美元）和人口总数。ii) 使用TreeMap<K,V>进行排序，通过Comparator接口和Comparable接口两种方式实现。  请自行查找中国、美国、德国、法国、日本、英国、意大利、韩国、印度、泰国这10个国家的GDP和人口信息，将这10个国家的信息添加到TreeMap中，并输出排序后的国家信息。  先创建Country类，包含name、gdp2023和population三个成员变量，用于初始化成员变量的有参构造器，用来返回私有成员变量gdp2023的getter方法，以及重写后的toString方法用来格式化输出国家信息。  接着在测试类的main方法中创建国家列表Countries，包含十个国家的GDP和人口信息。  然后使用TreeMap<K,V>进行排序，先通过Comparator接口的方式进行实现。在创建TreeMap对象的时候，可以通过Comparator匿名类的形式来自定义比较器。Comparator接口要求实现compare方法，该方法接收两个元素a、b并比较大小，a小于b返回负数，相等返回0，大于返回正数。TreeMap会根据比较结果对Key进行排序。因此重写compare方法只需传入两个double类型的参数来表示Country对象的gdp2023属性，再将差作为返回值即可，这样就实现了根据GDP从小到大排序。  然后将countries列表中的所有country对象以gdp2023属性值作为key，country本身作为value添加到TreeMap中即可，TreeMap内部会根据key来排序。最后输出排序后的列表。  运行程序，成功通过Comparator接口的方式实现了排序。  接着通过Comparable接口的方式进行实现。使Country类实现Comparable接口，并重写compareTo方法来自定义排序规则。注意该方法传入的参数为Country对象，因为需要排序的是Country对象，方法内调用getter方法来比较GDP。  在main方法中创建TreeMap，以Country对象作为key，并添加10个国家的信息，TreeSet内部会根据Key调用重写后的compareTo方法进行排序。最后遍历TreeMap的keySet集合，输出排序后的国家列表。  运行程序，成功通过Comparable接口的方式实现了排序。  (4) 编写Java应用程序。新建一个ArrayList对象、一个LinkedList对象，一个HashSet对象、一个LinkedHashSet对象、以及一个TreeSet对象。每个对象都包含1到50000的所有整数，顺序随机。现进行如下测试：i) 产生10000个随机整数，每个对象调用contains方法判断这些整数是否在该对象中；ii）计算每个对象执行这10000次操作所需要的时间，并进行对比分析。  在CollectionTest类中先定义常用MAX\_NUM和TEST\_NUM分别用来每个集合容纳的整数数量以及测试时操作的执行数量。  创建fillCollection方法用来填充集合对象。为实现每个对象包含1到50000的所有整数，顺序随机，先创建一个列表，并将1到50000的所有整数按顺序添加至该列表中，接着调用Collections类的shuffle方法来打乱该列表的元素顺序，最后调用addAll将打乱后列表中的所有元素添加到集合中。  再创建testAndRecord方法用来测试不同类型集合执行时间的差异。开始时先将当前时间设为开始时间，接着生成10000个随机整数，测试对象调用Collection类的contains方法来逐个判断这些整数是否在该对象中。全部判断结束后，记录结束时间，并输出整个判断操作的执行时间，即结束时间与开始时间之差。  在main方法中，先创建不同类型的集合，包括ArrayList、 LinkedList、HashSet、LinkedHashSet以及TreeSet。将顺序随机的1到50000的所有整数填充到每个集合中，并调用testAndRecord来输出不同类型集合的执行性能差异。  运行程序，可以看到Set类型的对象执行contains方法的效率要远远高于List类型的对象，这是因为调用contains方法时，ArrayList和LinkedList会遍历整个集合来寻找元素，时间复杂度为O(n)；Set类型中HashSet与LinkedHashSet内部都是通过哈希表实现，查找元素时会先计算其哈希码，再定位到哈希表中指定位置，平均情况时间复杂度接近O(1)，而TreeSet使用红黑树实现，时间复杂度为 O(log n)。  ArrayList相比于LinkList执行效率高，是因为虽然二者都是遍历整个集合，但ArrayList底层是采用连续的内存地址进行存储，并通过索引来访问数组位置，由此就能够更好地利用缓存行来读取数据；而LinkList使用链表实现，各个节点存储地址分散，就需要频繁对内存进行访问，影响执行效率。  3、线程同步（30分，每题10分）  (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。（10分）  **程序1:**  **程序2:**  **程序3:**  **程序一执行结果：**  **执行结果分析：**  运行程序一时，会创建三个线程，每个线程执行不同的任务：一个打印100次字符a，另一个打印100次字符b，还有一个打印数字1到100。  从程序一执行结果可见，三个线程输出的结果都不连续。这是因为这三个线程优先级相同，JVM中的线程调度器会轮流调用这三个线程进行输出，使得三个线程并发执行。虽然输出的顺序会不同，但是每个线程最终都会完成指定的任务量，字符a会被打印100次，字符b也会被打印100次，而数字1到100则会按顺序各打印一次。  而反复运行程序10次，得到的结果均不相同是因为线程的执行取决于操作系统的线程调度策略与当前CPU的使用情况，无法预测哪个线程会在什么时候获取CPU来执行，导致了输出结果的不可预测性。  **程序二执行结果：**  **执行结果分析：**  运行程序二时，会创建一个固定大小为3的线程池，接着会将三个Runnable任务提交给该线程池来执行，分别是打印100次字符a，打印100次字符b和打印数字1到100。执行完这些任务后关闭该线程池。  从程序二的执行结果可以看出，三个线程输出的结果同样不连续。这是因为在线程池中的每个线程会执行各自的任务，互不影响。由于创建的线程池是一个固定线程池，即核心线程数等于最大线程数，且程序指定了该线程池中的线程数为3，因此加入的三个任务会并行执行，三个线程输出的结果会交织在一起。与程序一相同，每个线程最终都会完成指定的任务量。  而反复运行程序10次，得到的结果均不相同是因为线程的输出取决于线程调度器在每次运行时如何分配CPU时间给这些线程，输出结果具有不可预测性。  **程序三执行结果：**  **执行结果分析：**  程序三模拟了多线程环境下对一个共享资源balance的并发访问。该程序先创建了一个动态大小的线程池，再向其中提交了100个任务，每个任务都会调用Account类的deposit方法来增加余额balance。该方法中在读到原余额balance的值后，会先让程序休眠5秒，再将更新后的余额值赋给balance。  通过执行结果可以看出，理论上应该输出100的程序却只输出了不到10的数，这是由于deposit方法没有进行同步处理，多个线程同时访问时会导致数据竞争。同一时间，多个线程都读取到了相同的balance值，并在休眠后都尝试将新余额设置回去，这可能就会导致原先已经增加很多的balance又被更新回最初的值。  而运行多次输出的结果均不相同，一部分是因为线程调度情况的不确定，更多的是因为多线程竞争结果的不可预测性。  (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。  在Test类中先创建输出所需的数字、英文单词数字以及月份列表。  接着定义共享锁对象lock，用于两个线程的同步，确保它们能够正确地交替执行。  然后先编写数字输出线程。该线程继承了Runnable接口，先用synchronized关键字确保同一时间只有一个线程执行同步块中代码。在同步块中，线程每输出一个数字后判断该数字是否是列表最后一项，是的话直接结束，否的话通知其他线程（此处就是英文单词输出线程）停止等待，进入准备执行状态，然后调用wait等待其他线程的通知。  再编写英文单词输出线程。该线程一开始同样需要使用synchronized关键字，这样就保证了这两个线程共享一个lock，不会出现并发问题。同步代码块内也是每输出一个英文数字加月份，就通知其他线程停止等待，随后调用wait等待其他线程的通知。  需要注意的就是wait方法可能抛出InterruptedException，表示当前线程在等待过程中被其他线程中断。这里采用的处理方式就是捕获该异常并打印栈跟踪信息，以便记录错误并调试。  在main方法中，通过Thread类构造方法创建了这两个线程，并调用start方法来启动。  运行程序，成功按顺序输出了数字、英文数字和月份。  (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。  先定义银行账户类BankAccount，成员变量包括double类型的账户余额balance，再创建用于初始化账户余额的有参构造器以及用于获取余额的getter方法。  然后创建withdraw方法来模拟银行现金账户取款操作，使用syncrhonized关键字修饰，确保同一时间只有一个线程可以执行该方法，防止多个线程同时修改，造成账户余额混乱。该方法内在执行取款操作前先确保取款金额为正数，且账户余额大于取款金额。然后输出取款前后的账户余额，来确保没有出现并发问题。  在Test类中创建WithdrawTask类，实现Runnable接口，负责构建取款任务。成员变量包括BankAccount对象以及取款金额amount，并通过有参构造器进行初始化。然后重写run方法来调用withdraw方法执行取款操作。  在main方法中，先创建一个银行账户对象，初始余额为1000元。接着创建一个固定大小为5的线程池，用于模拟多个线程同时执行取款操作。然后就创建10个取款任务并提交至该线程池，每次取款的金额为当前次数\*100，这样能够检查是哪几个线程成功执行了取款操作。  提交完成后，调用shutdown方法来关闭线程池，该方法会确保所有已提交任务完成后线程池才完全关闭。最后等待所有任务完成后输出最终账户余额。  运行程序，可以看到共有3次取款操作执行成功，其他都会显示余额不足而取款失败，可见程序确保了银行现金账户取款的有效和一致。  而取500与400块的先后次序与代码中不一致是因为synchronized同步锁的存在，当一个线程占有CPU资源，正在执行同步块内容时，其他尝试调用这个synchronized方法的线程会被阻塞，排队等待获取CPU资源，而CPU调度线程的顺序则具有不确定性。  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。（10）  **说明：**  （1）本次实验课作业满分为100分。  （2）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（含抄袭往届同学的作业）。**  （7）延迟提交将酌情扣分；期末考试阶段补交无效。 |

|  |
| --- |
| ++++++++++++++++++++++++++++++++++++++++++++++++++++++  **其他（例如感想、建议等等）。**  本次学习中，我学习了Java常见集合类和线程的相关知识，并通过实际编程联系加深了对这些概念的理解。实验过程中，通过一道排序题理解了Comparable与Comparator在使用上的区别与各自使用的场景；通过比较Collections类中ArrayList、LinkedList、 HashSet、LinkedHashSet、TreeSet查找数据时的速度，加深了对这些集合类的认识，也初步了解了这些类底层的数据结构和原理；还通过实际生活中的银行取款场景了解了并发编程的复杂性和重要性，同时也提醒了我在以后的开发时要格外注意多线程情况下数据的一致性问题。总的来说，这次实验不仅巩固了我的理论知识，还提高了我的实际编程能力，让我对Java语言有了更全面的认识。 |

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：  2024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。