**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1 基础知识、基本类型和类的初级应用**

**学院： 计算机与软件学院**

**专业： 软件工程**

**指导教师： 蔡树彬**

**报告人： 江浩 学号： 2022150083 班级： 软工2A**

**实验时间： 2024年9月3日- 9月22日**

**实验报告提交时间： 2024年9月22日**

**教务部制**

|  |
| --- |
| **实验目的与要求：**  **实验目的：**   1. 学习Java基础语法，包括数据类型、运算符、表达式和数组等相关内容。 2. 掌握Java基础语法，能准确回答相关问题并完成基础语法实验。 3. 能运用Java基础语法，编程解决基础算法问题。   **实验要求：**   1. 观看优课上Java程序设计课程视频的第1、2章 2. 完成优课上Java程序设计课程第1、2章的课后习题 3. 完成蓝桥云课上Java简明教程的基础语法（上、下）实验 4. 完成IntelliJ IDEA内置的“学习Java的IDE功能”课程 5. 使用Java编程解决“汉诺塔”问题    1. 递归（必做）    2. 优化（可选）    3. 注意计算并输出解决问题用时（可放大n） 6. 使用Java编程解决下列2个题目    1. “学生排队”问题   体育老师小明要将自己班上的学生按顺序排队。他首先让学生按学号从小到大的顺序排成一排，学号小的排在前面，然后进行多次调整。一次调整小明可能让一位同学出队，向前或者向后移动一段距离后再插入队列。  例如，下面给出了一组移动的例子，例子中学生的人数为8人。  0）初始队列中学生的学号依次为1, 2, 3, 4, 5, 6, 7, 8；  1）第一次调整，命令为“3号同学向后移动2”，表示3号同学出队，向后移动2名同学的距离，再插入到队列中，新队列中学生的学号依次为1, 2, 4, 5, 3, 6, 7, 8；  2）第二次调整，命令为“8号同学向前移动3”，表示8号同学出队，向前移动3名同学的距离，再插入到队列中，新队列中学生的学号依次为1, 2, 4, 5, 8, 3, 6, 7；  3）第三次调整，命令为“3号同学向前移动2”，表示3号同学出队，向前移动2名同学的距离，再插入到队列中，新队列中学生的学号依次为1, 2, 4, 3, 5, 8, 6, 7。  小明记录了所有调整的过程，请问，最终从前向后所有学生的学号依次是多少？  请特别注意，上述移动过程中所涉及的号码指的是学号，而不是在队伍中的位置。在向后移动时，移动的距离不超过对应同学后面的人数，如果向后移动的距离正好等于对应同学后面的人数则该同学会移动到队列的最后面。在向前移动时，移动的距离不超过对应同学前面的人数，如果向前移动的距离正好等于对应同学前面的人数则该同学会移动到队列的最前面。  **输入格式**  输入的第一行包含一个整数n，表示学生的数量，学生的学号由1到n编号。  第二行包含一个整数m，表示调整的次数。  接下来m行，每行两个整数p, q，如果q为正，表示学号为p的同学向后移动q，如果q为负，表示学号为p的同学向前移动-q。  **输出格式**  输出一行，包含n个整数，相邻两个整数之间由一个空格分隔，表示最终从前向后所有学生的学号。  **样例输入**  8  3  3 2  8 -3  3 -2  **样例输出**  1 2 4 3 5 8 6 7   * 1. “学生成绩”问题   数据结构老师小红将班级内学生的姓名和考试分数分别在一个字符串数组studentNames和一个浮点数组studentScores。两个数组的长度均为 n，并且对于每个下标i，studentNames[i]和studentScores[i]表示第i个人的姓名和考试分数。现需要按考试分数降序返回对应的姓名数组studentNames，并且若考试分数相同，原下标小的同学姓名需要排在前面。  **输入格式**  输入的第一行包含一个整数n，表示学生的数量。  接下来n行，每行一个字符串，表示n个学生的姓名。  再接下来n行，每行一个浮点数，表示n个学生的考试成绩。  **输出格式**  输出n行，每行一个字符串，表示按学生成绩排序后的学生姓名  **样例输入**  4  张三  李四  王五  老六  70.5  75  80  80  ["王五","李四","张三"]  **样例输出**  王五  老六  李四  张三 |
| **实验过程：**   1. 优课上Java程序设计课程视频第1、2章观看完成的截图      1. 优课上Java程序设计课程第1、2章课后习题的完成情况及得分截图            1. 蓝桥云课上Java简明教程基础语法实验（上、下）完成情况截图        1. IntelliJ IDEA内置的“学习Java的IDE功能”课程完成情况截图 2. 使用Java编程解决“汉诺塔”问题（必做）及优化（可选）   核心逻辑如下：   * + 使用递归将 n-1 个盘子移动到辅助柱子，将第 n 个盘子移动到目标柱子，然后再将 n-1 个盘子从辅助柱子移动到目标柱子。   System.currentTimeMillis() 用于计算程序的执行时间       1. 学生排队问题的测试用例设计思路和具体用例设计结果   设计思路：首先关注边界情况，在向后移动时，移动的距离不超过对应同学后面的人数，如果向后移动的距离正好等于对应同学后面的人数则该同学会移动到队列的最后面。在向前移动时，移动的距离不超过对应同学前面的人数，如果向前移动的距离正好等于对应同学前面的人数则该同学会移动到队列的最前面。然后按照题目所给操作流程和思路展开逻辑  具体用例设计结果：参照题目所给样例进行边界化处理：  当一共3个同学，移动1次，例如：3号向后移动一格，预期结果应为：1 2 3  实际结果如图：    符合预期。  当出现将不存在的同学进行移动时系统不会产生报错，默认为不进行任何移动：  例如一共3个同学，移动1次，将4号向后移动一格，不做任何变化。     1. 学生排队问题的解题思路（使用流程图或伪代码）和核心代码说明   核心代码说明如下：import java.util.ArrayList; import java.util.Scanner;  public class StudentMove {  public static void main(String[] args) {  Scanner in = new Scanner(System.in);  int n = in.nextInt(); *// 学生数量* int m = in.nextInt(); *// 移动指令数量* int[][] moves = new int[m][2]; *// 存放要移动的学生和移动的距离   // 输入移动指令* for (int i = 0; i < m; i++) {  for (int j = 0; j < 2; j++) {  moves[i][j] = in.nextInt();  }  }   *// 初始化学生队列* ArrayList<Integer> queue = new ArrayList<>();  for (int i = 1; i <= n; i++) {  queue.add(i); *// 学生的编号从1到n* }   *// 处理每个移动指令* for (int i = 0; i < m; i++) {  int studentNumber = moves[i][0]; *// 要移动的学生编号* int distance = moves[i][1]; *// 移动的距离   // 找到要移动的学生的当前索引* int currentIndex = queue.indexOf(studentNumber);  if (currentIndex != -1) {  queue.remove(currentIndex); *// 移除该学生* int newIndex = currentIndex + distance; *// 计算新位置* newIndex = Math.*max*(0, Math.*min*(newIndex, queue.size())); *// 确保新位置合法* queue.add(newIndex, studentNumber); *// 插入到新位置* }  }   *// 输出最终的队伍状态* for (int student : queue) {  System.out.print(student + " ");  }  } }   1. 学生得分问题的测试用例设计思路和具体用例设计结果   设计思路：按照题目要求先建立学生类，包括姓名和成绩，主函数中通过创建一个列表来存放学生信息，具体实现逻辑是按照成绩降序排序，边界条件是当成绩相同时，按照原索引进行升序排序。  具体用例设计，考虑到条件中当成绩相同时，按照原索引进行升序排序，我选取了5名成绩完全相同的同学测试，理论上他们的排序会和原索引一摸一样。  如图：测试通过。   1. 学生得分问题的解题思路（使用流程图或伪代码）和核心代码说明   import java.util.\*; // 导入java.util包中的所有类，用于使用Scanner和集合类  public class StudentScores {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in); // 创建Scanner对象用于从控制台读取输入  // 输入学生数量  int n = Integer.parseInt(scanner.nextLine()); // 从控制台读取一行输入，并解析为整数，表示学生数量  String[] studentNames = new String[n]; // 创建一个字符串数组用于存储学生姓名  float[] studentScores = new float[n]; // 创建一个浮点数组用于存储学生成绩  // 输入学生姓名  for (int i = 0; i < n; i++) {  studentNames[i] = scanner.nextLine(); // 逐行读取学生姓名并存储到数组中  }  // 输入学生成绩  for (int i = 0; i < n; i++) {  studentScores[i] = Float.parseFloat(scanner.nextLine()); // 逐行读取学生成绩并存储到数组中  }  // 创建一个列表来存放学生信息  List<Student> students = new ArrayList<>(); // 创建一个ArrayList存储Student对象  for (int i = 0; i < n; i++) {  students.add(new Student(studentNames[i], studentScores[i])); // 将每个学生的姓名和成绩封装成Student对象并添加到列表中  }  // 按成绩降序排序，若成绩相同则按原索引升序  students.sort((s1, s2) -> {  if (Float.compare(s2.score, s1.score) == 0) { // 如果成绩相同  return Integer.compare(s1.index, s2.index); // 按原始索引升序排列  }  return Float.compare(s2.score, s1.score); // 否则按成绩降序排列  });  // 输出排序后的学生姓名  for (Student student : students) {  System.out.println(student.name); // 打印排序后的学生姓名  }  }  // 学生类  static class Student {  String name; // 学生姓名  float score; // 学生成绩  int index; // 学生在输入中的原始索引  Student(String name, float score) { // 构造函数  this.name = name; // 初始化姓名  this.score = score; // 初始化成绩  this.index = indexCounter++; // 设置原始索引并递增计数器  }  private static int indexCounter = 0; // 静态变量保持索引，从0开始  }  } |
| **实验结论：**   1. 汉诺塔问题的优化思考和分析（可选） 2. 学生排队问题是否通过自行设计的测试用例？   通过   1. 学生得分问题是否通过自行设计的测试用例？   通过   1. 回顾整个实验过程中，你是否有遇到什么困难，你是如何解决的？   最大的困难是对于学生排队问题的边界条件处理上，处理方案是对所有的可能出现的边界异常罗列出来后进行一一测试。   1. 做完实验后，你最大的收获是什么？   对**ArrayList实现直接按索引插入使用更加熟练了** |
| **心得体会：**  通过本次实验，你是否能够使用Java语言解决一些简单的编程问题？是否还有其他的感想和建议？  可以。目前没有。 |
| 指导教师批阅意见：  成绩评定：  指导教师签字：蔡树彬  2024年9月2日 |