|  |
| --- |
| 一、实验目的   1. 本次实验的目的是掌握 Java 企业级应用开发工具的使用方法，掌握 Java   面向对象程序编程技术，掌握常用工具类的使用。理解和使用设计模式。   1. 设计开发完成一个基于命令行的软件。 2. 抄袭计 0 分。 |
| 二、实验项目内容  设计开发完成基于命令行的学生成绩管理系统。要求如下：  1、创建类实现基本对象和他们关系的管理。包括学生、教学班、课程、成绩、教师等。学生至少包含学号、姓名、性别等信息。教学班至少包含教师、课程名字、总人数、教学班号、开课学期等信息。课程至少包含课程编号、课程名字等信息。教师至少包含教师编号、姓名等信息。可以根据自己的分析设计增加其他类。  2、随机生成学生，数量不少于 100。课程数量不少于 3 门。一个教学班有一个教师上一门课程，一个教学班的学生数量不少于 20。教师数量不少于 6 个。一门课至少有两个老师上课。每个学生至少选择 3 门课程。一个学生  在一个教学班上一门课，考试后取得一个成绩。一门课的成绩构成有 4 部分构成，包括平时成绩、期中考试、实验成绩和期末考试成绩，然后计算出综合成绩。自定义各项成绩的产生策略，均为整数。  3、给出一个主菜单，选择菜单项执行各项功能。例如执行一个菜单命令，可以生成一个教学班的所有学生的平时成绩。分阶段模拟教学过程，第一步，生成初始化数据，包括教师，学生、课程，教学班等；第二步，学生选课，自定义选课策略，为每门课程的教学班安排学生。第三步，获得平时成绩，期中成绩，实验成绩，期末成绩，最后计算综合成绩，要记录成绩取得的时间。  4、能够格式规范地显示一个教学班的学生的成绩，可以根据学号排序，可以根据成绩排序。可以统计学生各科、总成绩的分数段分布。可以通过名字或者学号查询一个学生的所有科目的成绩和总成绩。可以按照学号、各科成绩和总成绩对所有学生进行排名显示。  5、可以实现自己的扩展功能或自定义功能。注意操作使用的方便性，注意类和类之间的关系。充分利用继承，多态等特性，使用上抽象类，接口，注解，Lambda 表达式，泛型，内部类等设计元素，使用好集合类。注意程序的总执行流程和分支执行流程。注意设计思想的表达，注意优化代码结构，优化类的职责分工，注意使用设计模式。代码有注释。  6、在报告中注明自己的创新点、特色等。 |

|  |
| --- |
| 三、实验过程或算法（写明：1 创新点或特色、2 设计思想、3 设计模式的使用、4 程序的结构或者架构、5 功能关系图、6 类的说明和类之间的关系图、 7 程序主要执行流程图，8 最后是核心源代码、截图等）  **1 创新点或特色**   1. 成绩生成策略   课程成绩的生成中用到了双层正态分布。该策略中用到两个属性和一个正态分布生成器方法类。  首先，写一个正态分布生成器方法类GenerateGaussia，为其传递均值、标准差、上限和下限，就可以返回一个指定范围内的正态分布的随机数。  然后，为学生添加了一个int类型属性level（五分制），用来代表学生的优秀等级，用正态分布生成器来确定大小；为课程添加了一个int类型属性difficulty（五分制）,用来表示课程的难度。  最后，用level、difficulty来确定某学生的某课程成绩的分数的正态分布类型。比如设定期末成绩为均值为30+15\*level-5\*difficulty,标准差为2\*level/3，上限为100，下限为0的正态分布。  这样我们就通过正态分布的学生优秀等级和课程难度来影响成绩的正态分布的均值和标准差，使得成绩的分布更加真实且合理，而不是简单的随机数。   1. 老师修改成绩功能   添加了老师修改自己所教班级学生的所教课程成绩的功能。该功能会验证用户输入的教师编号，若验证通过，则可更改该老师所教学生的所教课程的平时成绩、期中成绩、实验成绩、期末成绩。并自动更新综合成绩。   1. 学分机制   每个课程添加了一个学分的属性，用来模拟大学的绩点模式。用学分加权计算学生的总成绩。   1. 仿照关系型数据库模式   学生、老师、课程、班级这四个主要实体集之间通过关系型数据库的主键外键来联系。并且数据的展示中大部分也是采用类似mysql的表格的形式来展示。   1. 数据的展示形式   对于用户查询的普通信息，我们采用类似mysql的表格形式来展示。对于成绩分数段的统计则采用彩色横向条形图的形式来展示。如下图       1. 代码包结构   采用了工厂模式、命令模式、单例模式等多种模式，把代码文件划分了成了五个包，entity, factory, command, tool, service。分别存放实体集、工厂、命令、常用工具、服务层的类代码。    **2 设计思想**   1. 模块化设计   将代码拆分成多个方法或方法类来分模块进行编写，提高代码的可读性和编写代码时的条理性。例如将管理系统的五个主功能分别拆分编写成方法，主函数只需根据用户的输入调用指定的功能的方法。功能中对实体集进行的各种具体操作又包装成命令类，功能方法只需调用命令类的端口。例如下面的核查老师身份的命令：     1. 封装   数据和对数据的操作都封装起来，限制其他部分的代码直接访问，保证数据的安全性。四个主实体集都采用私有属性和公共方法，例如下图的教学班类实体集：     1. 继承和多态   创建了通用基类，使子类可共享或扩展父类的行为。例如学生，老师，课程，教学班这四个实体集都具有名字和编号两个属性，因此创建了一个AbstractNameAndId抽象类。或者例如命令类都有一个command父类接口。       1. 设计模式   使用了工厂模式、单例模式、命令模式等设计模式，根据设计模式的理念创建了大部分的类和划分了代码包结构   1. 可扩展性和可维护性   由于进行模块化编程，在新添加功能时，几乎不用更改之前的代码，只需创建新功能对应的方法或类就可以实现。代码出现bug时，也只需跟踪到出错的方法或类中，修改对应的方法或类即可，而不用关心一大串连在一起的代码。   1. 容错性   针对用户各种可能输入错误指令的情况，设计了指令错误提示的特定输出格式，并提示用户重新输入命令。例如下图：       1. 充分使用java设计元素   针对实验要求，充分使用上抽象类，接口，注解，Lambda 表达式，泛型，内部类等设计元素。  抽象类举例：    接口举例：    注解举例：    *由于本机java版本不够，lambda表达式不支持，因此该段代码在运行时先注释掉先使用下面那段等价代码。java版本够的话可将下面那段注释掉，使用上面的lambda表达式。*    泛型举例：    内部类举例:    **3 设计模式的使用**   1. 工厂模式   用途：用于创建实体集的对象，如学生、老师、课程、教学班。  实现：定义一个工厂接口interface Factory和一个实体集的通用基类abstract class AbstractNameAndId。再创建一系列继承AbstractNameAndId的实体集和对应的继承了Factory接口的工厂方法类。  例如：   1. 单例模式   用途：确保一个数据管理器类只有一个实例（数据管理器类用来生成、保存和清除所有实体集类对象），并减少内存开销。  实现：将数据管理器类的构造方法私有，并在类中创造它本身的一个静态实例，并提供一个静态方法，供外界获取它的静态实例。     1. 命令模式   用途：允许用户使不同的请求对客户端进行参数化。提高系统的扩展性，新命令可以很容易地添加到系统中。  实现：定义一个命令接口Command，将对实体集对象的数据的请求封装为一个继承了Command的对象。    **4 程序的结构或者架构**  该程序主要分为五个包entity, factory, command, tool, service，分别是存放实体集类、实体集的创建工厂类、命令类、常用工具类、服务层的类代码。  1.entity包存放学生、老师、课程、教学班等四个实体对象类以及他们的抽象类AbstractNameAndId，学生和老师的共同父类people。  2.factory包存放学生、老师、课程、教学班四个实体对象的工厂（用来创建它们的对象），以及他们的接口Factory。  3.service包存放服务端类，测试端类，以及数据管理器。服务端类即软件运行的主函数Main类，面向用户的，编写了服务端的代码逻辑。测试端类Test是开发者编写代码时用来测试方法类功能正常用的。Manage数据管理器类是负责生成所有实体对象（即学生，老师，课程，教学班）的数据，并保存这些数据，也可以清空这些数据，并且他是单例设计模式中的单例，负责在类和方法之间传送实体对象的数据。  4.command包存放实现服务端每个具体功能的命令类以及他们的接口Command。下图中command包里的命令类从上至下分别是：1更改学生成绩 2检查老师身份 3查询某班级基本信息 4清除所有实体数据 5生成所有实体数据 6对所有学生进行排名显示 7查询指定实体类的全部实体的基本信息 8查询指定班级所有学生及成绩信息 9查询指定学生的所有成绩信息 10查询所有学生或指定班级的各科、总成绩的分数段统计分布  5.tool包存放代码编写中常用到的工具方法类，如下图中从上至下是1生成条形图 2生成正态分布 3根据ID编号查询实体在实体数组中的索引 4打印表格 5按指定策略对学生进行排名 6查询指定学生的指定课程的成绩    **5 功能关系图**    **6 类的说明和类之间的关系图**  1.entity包存放学生、老师、课程、教学班等四个实体对象类以及他们的抽象类AbstractNameAndId，学生和老师的共同父类people。  2.factory包存放学生、老师、课程、教学班四个实体对象的工厂（用来创建它们的对象），以及他们的接口Factory。  3.service包存放服务端类，测试端类，以及数据管理器。服务端类即软件运行的主函数Main类，面向用户的，编写了服务端的代码逻辑。测试端类Test是开发者编写代码时用来测试方法类功能正常用的。Manage数据管理器类是负责生成所有实体对象（即学生，老师，课程，教学班）的数据，并保存这些数据，也可以清空这些数据，并且他是单例设计模式中的单例，负责在类和方法之间传送实体对象的数据。  4.下图中command包里的命令类从上至下分别是：1更改学生成绩 2检查老师身份 3查询某班级基本信息 4清除所有实体数据 5生成所有实体数据 6对所有学生进行排名显示 7查询指定实体类的全部实体的基本信息 8查询指定班级所有学生及成绩信息 9查询指定学生的所有成绩信息 10查询所有学生或指定班级的各科、总成绩的分数段统计分布  5.tool包下图中从上至下是1生成条形图 2生成正态分布 3根据ID编号查询实体在实体数组中的索引 4打印表格 5按指定策略对学生进行排名 6查询指定学生的指定课程的成绩      **7 程序主要执行流程图**    **8 最后是核心源代码、截图**  成绩生成代码：    服务端主函数代码：  public class Main {  public static void main(String[] args) {  Manage manager = Manage.*getInstance*();  int mainCommand = -1;  int dataExist = 0; *// 记录数据是否已经生成* Scanner scanner = new Scanner(System.*in*); *// 添加 Scanner 读取用户输入   welcomeWords*();*//欢迎词* do {  *mainMenu*();*//展示主功能菜单* System.*out*.print("请选择功能: ");  mainCommand = scanner.nextInt(); *// 读取用户输入的命令* switch (mainCommand) {  case 1: *// 生成数据* dataExist = *generateFunction*(dataExist,manager);*//生成数据，并返回1* break;  case 2: *// 查看数据  watchMenu*(dataExist,manager);*//查看功能菜单模块* break;  case 3: *// 管理数据  changeMenu*(dataExist,manager);*//管理功能菜单模块* break;  case 4: *// 清除数据* dataExist = *clearFunction*(dataExist,manager);*//清除数据，并返回0* break;  case 5: *// 退出  systemMessage*("退出成功！欢迎您下次使用！");  break;  default:  *exceptionMessage*("该命令无效！请您重新输入");  break;  }   } while (mainCommand != 5);   scanner.close(); *// 关闭 Scanner* }  数据查看方法代码：  public static void watchMenu(int dataExist,Manage manager) {  *//判断当前是否已有数据* if(dataExist==0)  {  *exceptionMessage*("抱歉，当前还未生成数据！请您先生成数据再进行该操作");  return;  }  *systemMessage*("欢迎进入数据查看模块！");  Scanner scan = new Scanner(System.*in*); *// 添加 Scanner 读取用户输入* int watchCommand;  do {  System.*out*.println("\n<数据查看功能菜单>");  System.*out*.println("1.按类查看基本信息");  System.*out*.println("2.查看指定教学班成绩");  System.*out*.println("3.查看指定学生成绩单");  System.*out*.println("4.查看学生总排名");  System.*out*.println("5.查看成绩分数段分布");  System.*out*.println("6.退出");  System.*out*.print("请选择功能: ");  watchCommand = scan.nextInt(); *// 读取用户输入的命令* int secondCommand=0,thirdCommand=0;*//二,三级指令* String nameSearch="";*//按姓名找学生用* switch (watchCommand) {  case 1: *// 按类查看基本信息  //菜单* System.*out*.println("\n<按类查看基本信息>");  System.*out*.println("1.学生类");  System.*out*.println("2.老师类");  System.*out*.println("3.课程类");  System.*out*.println("4.教学班类");  System.*out*.print("请输入你要查看的类：");  secondCommand=scan.nextInt();  *//判断错误输入* if(secondCommand<1||secondCommand>4)  {  *exceptionMessage*("输入不合法！请重试");  continue;  }  SearchBaseCommand searchBaseCommand=new SearchBaseCommand();  searchBaseCommand.execute(manager,secondCommand,0,0,"");  break;  case 2: *// 查看指定教学班成绩* System.*out*.print("\n请输入你要查看的班级编号：");  secondCommand=scan.nextInt();  *//判断班级是否存在* if(IdSearchIndex.*searchIndex*(manager.getClassrooms(),secondCommand)==-1)  {  *exceptionMessage*("该班级不存在！请重试");  continue;  }  System.*out*.print("请输入你要查看的排序方式（1为学号排序，2为课程成绩排序）：");  thirdCommand=scan.nextInt();  *//判断指令合法* if(thirdCommand!=1&&thirdCommand!=2)  {  *exceptionMessage*("该指令不合法！请重试");  continue;  }  SearchClassCommand searchClassCommand=new SearchClassCommand();  searchClassCommand.execute(manager,thirdCommand,secondCommand,0,"");  break;  case 3: *// 查看指定学生成绩单* System.*out*.print("\n请输入你的搜索方式（1为学号查找，2为姓名查找）：");  thirdCommand=scan.nextInt();  *//判断指令合法* if(thirdCommand==1)  {  System.*out*.print("\n请输入你要查找的学生ID：");  secondCommand=scan.nextInt();  *//判断学生id是否存在* if(IdSearchIndex.*searchIndex*(manager.getStudents(),secondCommand)==-1)  {  *exceptionMessage*("该学生不存在！请重试");  continue;  }  }  else if (thirdCommand==2)  {  System.*out*.print("\n请输入你要查找的学生姓名：");  nameSearch=scan.next();  }  else  {  *exceptionMessage*("该指令不合法！请重试");  continue;  }  SearchStudentCommand searchStudentCommand=new SearchStudentCommand();  int live=searchStudentCommand.execute(manager,thirdCommand,secondCommand,0,nameSearch);  if(live==0) *exceptionMessage*("该学生不存在！请重试");*//不存在该姓名的学生* break;  case 4: *// 查看学生总排名* System.*out*.print("\n请输入你要查看的排序方式（1为学号排序，2为课程成绩排序则，3为总成绩排序）：");  secondCommand=scan.nextInt();  if(secondCommand==2)  {  System.*out*.print("请输入你要指定的课程编号：");  thirdCommand=scan.nextInt();  }  RankingCommand rankingCommand=new RankingCommand();  rankingCommand.execute(manager,secondCommand,thirdCommand,0,"");  break;  case 5: *// 查看成绩分数段分布* System.*out*.print("\n请输入你要查看的分数段（1为全体学生总成绩及课程成绩的统计，2为指定教学班的成绩）:");  secondCommand=scan.nextInt();  if(secondCommand==1)  {  StatisticsCommand statisticsCommand=new StatisticsCommand();  statisticsCommand.execute(manager,1,0,0,"");  }  else if(secondCommand==2)  {  System.*out*.print("请输入你要查看的教学班编号:");  thirdCommand=scan.nextInt();  *//判断班级编号是否存在* if (IdSearchIndex.*searchIndex*(manager.getClassrooms(),thirdCommand)==-1)  {  *exceptionMessage*("该班级不存在！请重试");  continue;  }  StatisticsCommand statisticsCommand=new StatisticsCommand();  statisticsCommand.execute(manager,2,thirdCommand,0,"");  }  else  {  *exceptionMessage*("该指令不合法！请重试");  continue;  }  break;  case 6: *// 退出* break;  default:  *exceptionMessage*("该命令无效！请您重新输入");  break;  }   } while (watchCommand != 6);  *systemMessage*("已退出数据查看！"); } |
| 四、实验结果及分析和（或）源程序调试过程（界面截图和文字）、实验总结与体会（必写）  **实验结果**：   1. 进入系统，输入1生成数据      1. 查看学生类基本信息，正确      1. 老师类基本信息，正确     3.课程类基本信息，正确     1. 教学班类基本信息，正确     5.指定班成绩排序，正确    6.指定学生成绩，正确    7.总成绩排序，正确    8.分数段统计，正确    9.修改成绩正确        10.清除数据，正确    经过测试，所有功能均正确实现，实验完成！  由于功能太多，不好一一在报告中验证，演示视频中更完整的展示了全部功能的正确。  **实验总结与体会**：  因为这是第一次java实验，所以对设计模式、结构化编程的思想、实验报告中设计的阐述和图形的绘画不够熟悉，实践起来更困难一些。  但没有很难的算法需要实现。可以把注意力主要放在熟悉设计模式、结构化编程以及思考创新点上。代码编写过程中一大部分时间也是在设计代码的结构上。  大部分难点经过第一次实验的熟悉，在之后的实验中应该可以更顺畅地完成。 |