|  |
| --- |
| 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 Web编程技术；掌握常用Java后端开发框架的使用，例如SpringBoot等；理解所用框架的工作原理。 2. 设计开发完成一个基于Web的成绩管理系统。 3. 抄袭计0分。 |
| 二、实验项目内容  设计开发完成一个基于Web的成绩管理系统。要求如下：   1. 后端可以使用SpringBoot等框架，**也可以使用其他框架**。要求数据存储在数据库中。自由选择使用什么数据库。 2. 前端使用浏览器访问软件系统界面。**自由选择**前端界面技术。自由选择是否前后端分离架构。 3. 提供账号注册、登录、注销等功能。教师账号管理学生成绩，学生账号查看学生成绩。教务账号管理班级，课程等。 4. 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。 5. 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。 6. 提供合适的操作界面完成上述功能。可以录入成绩，可以批量产生成绩等；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。 7. 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，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，注意使用各种框架。代码有注释。 8. 在报告中注明自己的创新点、特色等。 9. 提交：（1）本实验报告，（2）源代码压缩文件zip，（3）软件演示的MP4视频，视频大小不超过40M，视频请在**QQ浏览器**测试能否正常播放。注意源代码加注释。注意文件名称的规范性。文件名：学号姓名4.docx，学号姓名4.zip, 学号姓名4.mp4。**三个文件分别提交**。 |
| 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功能组织图（思维导图）、6类的说明和类之间的关系图（UML规范的类图）、7程序主要执行流程图（符合行业规范），8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. 使用Springboot和Mybatis-plus，以及mysql数据库   用Springboot来构建项目结构，Mybatis-plus来完成对数据库的操作。Mysql建立了六个表格来完成对数据的管理。     1. 前后端分离架构   前端使用html，css和javascript的组合。共设计了8个前端页面。   1. 使用APIfox来管理后端接口，共实现了40个接口      1. 数据属性完备，且相关的不同属性的数据之间做好了同步   在实验要求基础上，添加了学生等级、课程难度、课程学分、课程选修总人数等属性，来实现更真实更贴合实际的成绩。  班级人数、课程选修人数这些属性之间也做好了同步，例如学生注销时会将班级人数和选修的课程人数减一，删除班级时也会讲课程的人数减去该班级人数。   1. 扩展功能   .对于新注册的学生，增加了一键选课功能，以防新用户学生没有成绩信息。  .为学生、老师、教务增加了退出登录功能，方便更换账号。  .为教务增加了增加和删除课程和班级的功能，以便更好地管理课程班级。  .为教务增加了一键生成所有数据，一键清除所有数据，一键选课，一键生成成绩等功能。方便对数据进行管理和对教学进行分阶段控制。   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用Chart.js在前端生成彩色条形统计图的形式来展示。如下图       1. 界面美观，操作便利   分别使用了三种视频背景          对批量生成成绩功能，使用了复选框的样式来实现    对数据进行修改和添加也简洁明了       1. 代码包结构   采用了工厂模式、命令模式、单例模式、数据访问对象模式、MVC模式等多种设计模式，把代码文件划分了成了8个包，共44个类。    **2设计思想**   1. 前后端分离架构   后端实现负责实现API接口，前端用html，css和js来设计网页结构，并用js发送和接收http请求来与后端端口连接。用apifox来便捷管理后端接口。   1. 模块化编程，重视代码的复用   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、编写代码时的条理性以及代码的可复用性。例如这次就由于前几次实验的模块化编程带来的代码复用性，使得一些代码可以直接借鉴或略微修改就应用到这次实验中。例如选课系统和成绩生成器。   1. 封装、继承和多态   数据和对数据的操作都封装起来，用private限制其他部分的代码直接访问类的各项属性，保证数据的安全性，并使用工厂类进行创建模型。例如下图：    应用工厂模式、命令模式时都先创建接口，再创建具体工厂和命令类来继承接口进而实现不同功能的方法。   1. 设计模式   使用了工厂模式、单例模式、命令模式、数据访问对象模式、MVC模式等多种设计模式，把代码文件划分了成了8个包，共44个类。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. 开放/关闭原则   项目增加功能的过程中也是重视开放/关闭原则。增加功能时对代码已经实现的功能的代码尽量不修改，而是在不更改现有代码的前提下进行扩展。尽量增加新的类和方法，而不是修改之前的代码。   1. 重视用户体验   对批量生成成绩功能，使用了复选框的样式来实现    对数据进行修改和添加也简洁明了      考虑到注销账户、删除课程、删除班级等操作是重要操作，添加了确认操作的环节，并使用了emoji表情。     1. 充分利用java特性和技术   重视对java各种技术和工具的使用，在实践中磨炼对java技术的掌握和理解。充分使用上抽象类，接口，注解，Lambda 表达式，泛型，内部类等设计元素以及springboot和Mybatis-plus框架。例如下面的举例  抽象类：    接口：    注解：    Lambda 表达式：    泛型：    内部类：    Springboot：    Mybatis-plus：    **3设计模式的使用**   1. 工厂模式   用途：更安全和方便地扩展对数据对象的初始化。  实现：创建了数据对象的接口和工厂的接口     1. 命令模式   用途：将请求封装为一个对象，允许用户使用不同的请求对客户端进行参数化。  实现：将对文件进行操作的命令都封装成命令类，继承命令接口。     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. MVC模式   用途：将应用程序分为三个核心组件：模型（Model）、视图（View）和控制器（Controller），以实现关注点分离。。  实现：模型在pojo包内实现与数据库连接的各实体对象的定义 ，视图在前端页面用html，css，js等来实现。控制器在Controller包里来接收前端的接口请求并完成后端对数据的操作。       1. 数据访问对象模式(DAO 模式)   用途：把低级的数据访问 API 或操作从高级的业务服务中分离出来。  实现：把对数据库实体对象的操作封装到DAO类，在此直接继承mybatis-plus框架的Mapper类，再在service包里具体封装方法      **4程序的结构或者架构**  代码文件划分了成了8个包，共44个类。其中BGradeSystemApplication.java是Springboot项目的启动器，controller是接收前端http请求的控制器包。具体类功能在下面树形图中进行解释    └─bgradesystem  │ BGradeSystemApplication.java Springboot项目启动器  │  ├─command 命令包  │ Command.java 命令接口  │ GradeRankCommand.java 按成绩排序的命令  │ IdRankCommand.java 按id排序的命令  │  ├─controller 接收前端http请求的控制器包  │ AllRankController.java 接收关于学生总排名的请求  │ ClassGradeController.java 接收关于班级成绩的请求  │ ClassroomController.java 接收关于班级的请求  │ CourseController.java 接收关于课程的请求  │ LoginController.java 接收关于登录的请求  │ ManagerController.java 接收关于教务的请求  │ StatisticsController.java 接收关于成绩统计分布的请求  │ StudentController.java 接收关于学生的请求  │ StudentGradeController.java 接收关于学生成绩的请求  │ TeacherController.java 接收关于老师的请求  │  ├─dao 数据访问对象继承Mapper  │ ClassroomDao.java 班级的数据访问对象  │ CourseDao.java 课程的数据访问对象  │ GradeDao.java 成绩的数据访问对象  │ ManagerDao.java 教务的数据访问对象  │ StudentDao.java 学生的数据访问对象  │ TeacherDao.java 老师的数据访问对象  │  ├─factory 工厂包  │ ClassroomFactory.java 班级的工厂  │ CourseFactory.java 课程的工厂  │ Factory.java 工厂接口  │ GradeFactory.java 成绩的工厂  │ ManagerFactory.java 教务的工厂  │ StudentFactory.java 学生的工厂  │ TeacherFactory.java 老师的工厂  │  ├─pojo 数据库实体对象  │ Classroom.java 班级  │ Course.java 课程  │ Grade.java 成绩  │ Manager.java 教务  │ People.java 人抽象类  │ Student.java 学生  │ Teacher.java 老师  │  ├─service 服务层（包装好数据库操作）  │ ClassroomService.java 班级服务  │ CourseService.java 课程服务  │ GradeService.java 成绩服务  │ ManagerService.java 教务服务  │ StudentService.java 学生服务  │ TeacherService.java 老师服务  │  ├─singleton 单例  │ AllData.java 数据管理器  │  └─tool  GenerateGaussian.java 正态分布生成器  IdSearchIndex.java 数组中id找索引  Message.java 返回前端http的message构造器  **5功能组织图（思维导图）**    **6类的说明和类之间的关系图（UML规范的类图**）  类的具体功能说明，已在程序架构部分进行阐述。  由于类的数量众多，画在一起依赖表示线太密太乱，在此分开进行说明。  对于六个基础实体集和工厂    对于单个实体，与其对应的dao、factory、service、Controller的关系（在此以学生举例，其他五个实体与之一致）。  Student由StudentFactory创建,由StudentDao继承Mapper接口管理与其对应的数据库表。  Service层的StudentService使用StudentFactory和StudentDao来对student的数据库操作进行封装。  最后由StudentController来接收前端http请求，并用StudentService对student的数据库进行操作    AllData类通过使用六个实体的Service类来对全部数据进行管理和生成。    **7程序主要执行流程图（符合行业规范）**    **8是核心源代码、截图等）**  学生的controller      班级成绩的controller    学生的service层      Student的DAO    Student的实体对象 |
| 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）  **实验总结与体会**：  这次的实验相比上次用jsp和servlet写网站，项目的配置过程顺利了很多，没有上次那么多的小细节问题，不至于因为一个配置问题卡一天。主要原因也是springboot本身就为我们简化了配置操作。  也因为springboot目前比较流行，网上的样例很多。因此项目的架构设计上也比较顺利。借鉴了这种分pojo层、dao层、service层、controller层的四层架构设计方式。没有遇到太大的困难。  这次的难点主要感觉是前后端的API接口的对接问题。采用前后端分离架构时，要结合html和javascript来在前端发送和接受并处理后端的数据。由于我在暑假实训时负责了主要的前端代码的编写工作才得以在此难点上比较顺利，但还是由于繁琐，花费了很大部分时间。对于没有接触过前端技术和对javascript不够了解充分的同学来说，这个难点可能会困扰更多时间，也许要比在后端花费的时间要多得多。  也庆幸我在暑假的实训中选择了前端的任务，让我不至于只接触后端技术而对前后端的对接的过程一窍不通，也让我有能力仅凭自己从前端到后端构建出一个界面简洁美观，后端功能完善的网站项目，说不准还有全栈工程师的潜力。  **实验结果分析：**  由于演示视频中已经将功能更好地展示完善，并且功能的正确性大多要考动态操作来证明，不方便使用截图。因此在此主要展示前端界面的正确性和前端接口获取信息的正确性。  登录页面    注册页面    学生端    教师端    教务端      教务端成绩分布统计：    教务端学生总排名 |