

成绩：_____

南京信息工程大学

数据库管理系统 课程设计

题 目： 学生信息管理系统的设计与实现

学 院： 计算机与软件学院

专 业： 物联网工程

学生姓名： 张逸飞

学 号： 20171375029

指导教师： 张群

年 月 日

目录

一、 前言	1
二、 需求分析.....	1
三、 概要设计.....	2
1. 项目代码的设计.....	2
2. 数据库的设计.....	2
3. 整体项目的类框架.....	3
四、 详细设计.....	4
1. 设计思路.....	4
2. 部分类的详细设计.....	5
3. 数据库的详细设计.....	6
五、 部分代码及注释.....	7
六、 运行过程.....	11
七、 总结.....	12

一、前言

为了充分应用数据库系统理论的相关知识，将其应用于项目实践中，因而设计并实现了一个项目：学生信息管理系统。选择 Java 语言用于实现项目。

对于学生成绩管理系统，应具有以下功能：学生成绩查询功能，包括特定学生成绩查询；学生成绩排序功能；求各科成绩平均值的功能；学生选课情况查询功能；增删学生信息的功能；修改学生信息的功能等。使用数据库维护两张表，一是学生成绩表，二是学生选课情况表。

环境：系统环境为 Windows，Java 环境为 JDK 1.8，IDE 选用 Eclipse。

前端实现可以使用 Java Swing 提供的 GUI 编程功能，主要涉及 JButton、JLabel、JPanel、JTable、JText、JComboBox、JMenu、JMenuBar 等类。

数据库使用 SQL Server。主要在 cmd 中执行命令，辅助以可视化数据库管理软件 Microsoft SQL Server Management Studio。

二、需求分析

学生信息管理系统：

学生信息管理系统需要解决的问题如下：方便管理者进行数据的录入、修改、删除、查找。录入的信息应该包括学生的学号、各科成绩、班级信息、选课情况、老师的授课班级、教授的课程等，修改应该能够修改学生的全部属性；删除特定学生的所有信息；查找要能够按指定条件查找；修改教师授课情况。查找的实现比较关键，很容易可以想到，删除和修改的操作都是在查找到学生的前提下进行修改。

由上述需求分析可以得知该项目应该具有的功能：按照学号查找的查询功能，删除信息功能，修改学生信息的功能，增添学生信息的功能，修改教师信息的功能，增加教师教授的班级和

课程，删除教师信息。学生信息应该包含以下属性：学号、班级、姓名、各科成绩、总分、各科选课情况等。教师信息应该包括：授课班级，教授课程等。除此以外应该实现按总分排序，求班级各科平均值等常用功能。

三、 概要设计

项目代码的设计：


本次整体项目设计采用 MVC 模式，即模型（modle）—视图（view）—控制器（controller）模式。用于将业务逻辑、数据、界面显示分离。模型是数据模型和数据访问模型；视图用于控制用户界面的样式和系统和用户的交互行为；控制器是将视图的任务分配给特定的模型解决。

数据库的设计：

数据库设计采用了四张表格。与学生相关的信息：学号、班级、姓名、各科成绩、总分、各科选课情况。与教师相关的信息：教师姓名、教授班级、教授课程。根据范式理论，为了避免使用中出现数据冗余、插入删除异常等状况，将学生信息划分为三张表单。第一张为 profile，包含学生基本信息：学号班级、姓名；第二张为 course，包含学生选课信息：学号、各科选课情况；第三章为 grade，包含学生课程成绩：学号、各科成绩。

表单属性具体如图所示：

profile 表单：

	列名	数据类型	允许 Null 值
	学号	nchar(20)	<input type="checkbox"/>
	姓名	nchar(10)	<input type="checkbox"/>
	班级	nchar(10)	<input checked="" type="checkbox"/>

grade 表单:

	列名	数据类型	允许 Null 值
🔑	学号	nchar(20)	<input type="checkbox"/>
	语文	int	<input checked="" type="checkbox"/>
	数学	int	<input checked="" type="checkbox"/>
	英语	int	<input checked="" type="checkbox"/>

course 表单:

	列名	数据类型	允许 Null 值
🔑	学号	nchar(20)	<input type="checkbox"/>
	地理	bit	<input checked="" type="checkbox"/>
	人文	bit	<input checked="" type="checkbox"/>
	历史	bit	<input checked="" type="checkbox"/>
	政治	bit	<input checked="" type="checkbox"/>
	物理	bit	<input checked="" type="checkbox"/>
	化学	bit	<input checked="" type="checkbox"/>
	生物	bit	<input checked="" type="checkbox"/>

teacher 表单:

🔑	班级	nchar(10)	<input type="checkbox"/>
🔑	课程	nchar(10)	<input type="checkbox"/>
	任课教师	nchar(10)	<input checked="" type="checkbox"/>

整体项目的类框架:

模型部分 (modle):

- | | |
|----------------|-----------------|
| courseModle 类 | course 表单的数据模型 |
| profileModle 类 | profile 表单的数据模型 |
| gradeModle 类 | grade 表单的数据模型 |
| teacherModle 类 | teacher 表单的数据模型 |

Student 类 学生的整体数据模型
userStore 类 用户登陆的数据

视图部分 (View):

DeleteButton 类 包含删除事件
ErrorPopup 类 错误触发事件
InsertButton 类 插入事件
Login 类 登陆事件
OperateScreen 类 主界面
SearchButton 类 查找事件
SelectWindow 类 选择界面
ShowButton 类 显示界面
Universe 类 View 中的通用属性

控制器部分 (controller):

courseControl 类 course 表单的相关事件处理
gradeControl 类 grade 表单的相关事件处理
profileControl 类 profile 表单的相关事件处理
StudentControl 类 Student 模型的事件处理
teacherControl 类 teacher 表单的事件处理

四、详细设计

设计思路。

数据模型部分:

数据模型应该包含相关表单的所有数据和关于数据的操作。由于采用的 MVC 设计模式，数据模型应该只关乎于数据的形式，而不用操心与数据库的交互。数据模型应该为 controller 部分提供数据操作的接口。如对于 profileModle 类，它应该包含私有属性：学号、姓名、班级，公有方法：getID(), getName(), getClassNumber()。

控制器部分:

控制器的基本操作对象是数据模型。它提供与数据库交互的相关方法。其中有一个比较特殊的类：GetConnection 类。该类主要用于控制与数据库连接的建立和关闭，以保证资源的

及时释放。控制器的所有类的方法应该是静态方法，原因在于控制器部分并不需要显示创建对象，其方法应该可以直接调用，同时可以为 View 部分的提供更加方便的服务。值得注意的是，控制器作为三层中的中间一层，它对上（view 部分）屏蔽了具体的数据模型。这样当数据属性发生改变的时候，只要控制器的接口不变，View 部分就不需要改变。

视图部分：

视图部分用于显示图形界面，并提供人机交互的事件监听。它应该具备能为相同数据模型提供不同显示方案的能力。它于数据模型部分不直接相关，调用的是控制器的接口来与数据库进行交互。

部分类的详细设计：

Login 类：

根据数据库管理系统的权限特点，应该设计一个登陆界面验证用户的身份信息，并且输入的账号、密码等信息应传递给数据库管理系统验证并授权。Login 是初始的界面，在登陆成功后应该关闭当前界面，进入下个界面以推进事务的进行。

DeleteButton 类：

单击删除按钮应该弹出一个新的删除界面用于数据的删除。在删除按钮类设计的过程中，需要额外设计一个删除界面，并设置鼠标单击的事件响应，打开删除界面。

SearchButton 类：

单击查找按钮应该出现查找界面。对于学生信息管理系统的查找界面提供按学号查找的功能，具体实现方式是提供文本框，使用户输入学号，单击确定按钮后出现新的页面显示结果。

InsertButton 类：

插入按钮事件应该弹出插入界面。插入界面提供一张空白表格供用户输入信息，单击确认按钮即可插入所输信息。

gradeModle 类：

一个 gradeModle 对象可以储存 grade 表单中某一行的数据。表现为某一行的代码实例。提供 get 方法返回某属性值。

gradeControl 类：

gradeControl 内部具有 execute 的私有方法，给它传递 sql

命令（以 String 的形式）后可以与数据库进行交互。每次交互进行一次连接，完成交互后释放连接。与数据库交互的方式：更新数据、插入新数据、查找数据等。

GetConnection 类：

GetConnection 是一个比较特殊的控制器类，因为它与数据模型无关。GetConnection 类为其他控制器类提供连接的建立和关闭服务。使用 JDBC 作为驱动程序。

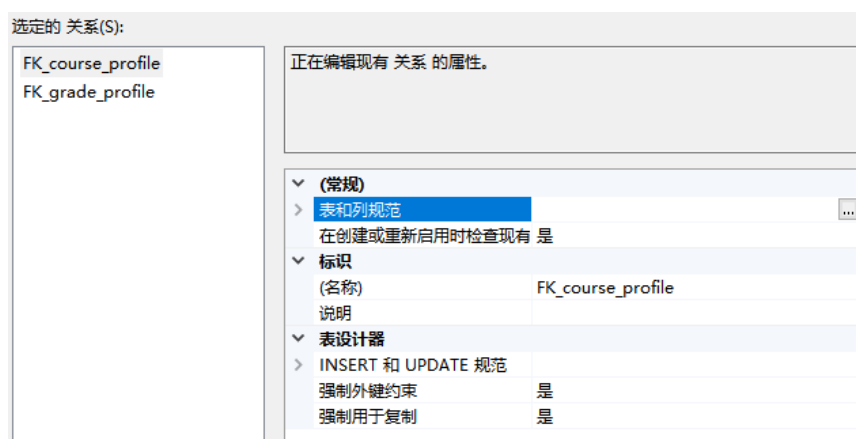
数据库的详细设计：

主键：

很容易可以看出对于学生信息而言，学号是其唯一标志。我们将学号作为 profile、course、grade 的唯一主键且不允许默认 null 值。另外，考虑到实际情况：一个班级的某个课程只有一个老师任教。所以 teacher 表单的主键设为班级和课程。

外键：

此次设计考虑到 profile、course、grade 三表的关系，为了避免产生异常或者冗余无效的信息，我们对其插入数据的顺序加以限制。我们以 profile 为主。当 profile 表单中不存在某个学号时，不能进一步对该学号 course、grade 表单的信息进行插入。根据这一要求我们对三表进行外键设置。

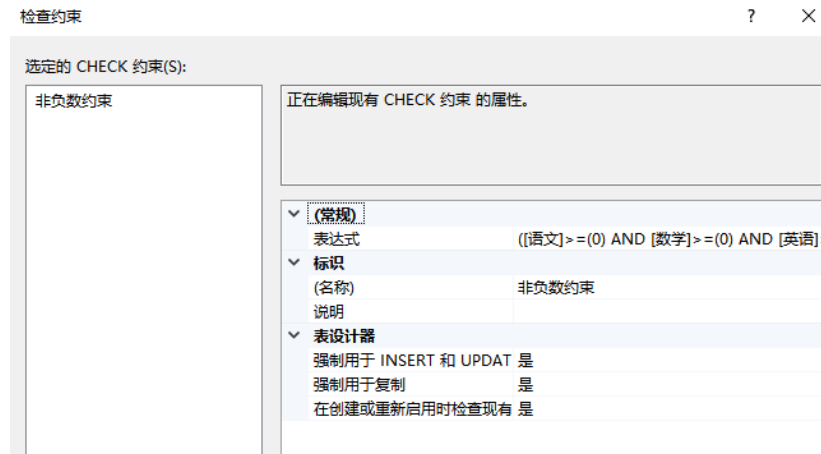


约束：

非负数约束：即所有成绩应该大于等于 0。约束表达式为：
语文>=0 AND 数学>=0 AND 英语>=0。

最大值约束：即所有成绩应该小于等于 100。约束表达式

为：语文<=100 AND 数学<=100 AND 英语<=100



权限管理：

设有学生、教师、管理员三种账号。

学生账号：可以使用查看功能，无法修改、删除、增加。

教师账号：可以使用查看、修改、删除、增加学生信息，查看教师信息，无法修改教师信息。

管理员账号：可以查看、修改、删除、增加学生、教师信息。

五、 部分代码及注释

由于模型类之间和控制器类之间的设计很类似，因此以 gradeModle 和 gradeControl 为代表。并附上 GetConnection 的相关代码。由于用户图形界面主要使用 Java 语言的相关知识，与数据库的关系不是很大，因此略过 View 部分的代码。

gradeModle 的代码：

```
public class gradeModle {  
    //grade表单的属性  
    private String ID;  
    private int ChGrades;  
    private int MathGrades;  
    private int EngGrades;  
    private int SumGrades;  
    //提供两种构造方法  
    public gradeModle() {}  
    public gradeModle(String id,int c,int m,int e,int s) {
```

```

        ID = id; ChGrades = c; MathGrades = m; EngGrades = e;
        SumGrades = s;
    }
    //提供get方法
    public String getID() {
        return ID;
    }
    public int getChineseGrades() {
        return ChGrades;
    }
    public int getMathsGrades() {
        return MathGrades;
    }
    public int getEnglishGrades() {
        return EngGrades;
    }
    public int getSumGrades() {
        return SumGrades;
    }
}

```

gradeContro 的代码:

```

public class gradeControl {
    private static Connection conn;
    private static PreparedStatement pstmt;
    private static ResultSet rs;
    private static GetConnection connection = new GetConnection();
    //使用java.sql中的excute来执行sql语句与数据库交互
    private static void update(String sql) {
        try {
            pstmt = conn.prepareStatement(sql);
            pstmt.execute();
        } catch (SQLException se) {
            new ErrorPopup("Update failed");
            se.printStackTrace();
        } finally {
            connection.closed(pstmt, conn);
        }
    }
    //使用executeQuery来执行sql语句，并返回结果集
    private static ResultSet select(String sql) {
        conn = connection.getConnection();
    }
}

```

```

PreparedStatement _pstm = null;
ResultSet _rs;
try {
    _pstm = conn.prepareStatement(sql);
    _rs = pstm.executeQuery();
    return _rs;
}catch(SQLException se) {
    new ErrorPopup("select error");
    se.printStackTrace();
}finally {
    connection.closed(_pstm,conn);
}
return null;
}
//更新数据
public static void updateChinese(int newgrade,String ID) {
    String sql = "update grade set 语文='"+newgrade+"' where 学号='"+ID+"'";
    update(sql);
}
//之后还有updateMaths,updateEnglish等方法，因为和updateChinese类似，略
//通过学号查询，返回数据模型
public static gradeModle selectWithID(String ID) {
    String sql = "select * from grade where 学号='"+ID+"'";
    rs = select(sql);
    gradeModle grade = null;
    try {
        while(rs.next()) {
            String id = rs.getString("学号");
            int ch = rs.getInt("语文");
            int ma = rs.getInt("数学");
            int en = rs.getInt("英语");
            int sum = ch+ma+en;
            grade = new gradeModle(id,ch,ma,en,sum);
        }
    }catch(SQLException se) {
        new ErrorPopup("Get grade info failed!");
    }
    return grade;
}
//选择表单中所有数据
public static List<gradeModle> selectAll(){
    String sql = "select * from grade";
    rs = select(sql);

```

```

List<gradeModle> list = new ArrayList<gradeModle>();
try {
    while(rs.next()) {
        String id = rs.getString("学号");
        int ch = rs.getInt("语文");
        int ma = rs.getInt("数学");
        int en = rs.getInt("英语");
        int sum = ch+ma+en;
        gradeModle grade = new gradeModle(id,ch,ma,en,sum);
        list.add(grade);
    }
} catch(SQLException se) {
    new ErrorPopup("Get grade info failed!");
}
return list;
}

//插入数据
public static void insert(String id,int ch,int ma,int en) {
    String sql = "insert into grade values
('"+id+"','"+ch+"','"+ma+"','"+en+"')";
    update(sql);
}

}

}

GetConnection 类代码:
public class GetConnection {
    private String classname =
"com.microsoft.sqlserver.jdbc.SQLServerDriver";
    private String url =
"jdbc:sqlserver://DESKTOPYI\\SQLEXPRESS:1433;DatabaseName=StuData";
    private String username = "sa";
    private String passwd = "passwd";
    public Connection getConnection() {
        Connection conn;
        try {
            //装载JDBC驱动
            Class.forName(classname);
            conn = DriverManager.getConnection(url, username, passwd);
        } catch(Exception e) {
            new ErrorPopup("Connection Error!");
            conn = null;
            e.printStackTrace();
        }
        return conn;
    }
}

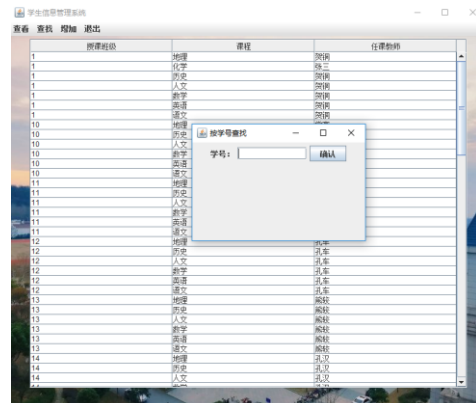
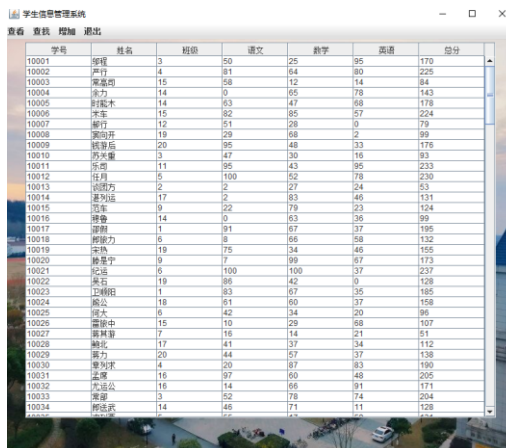
```

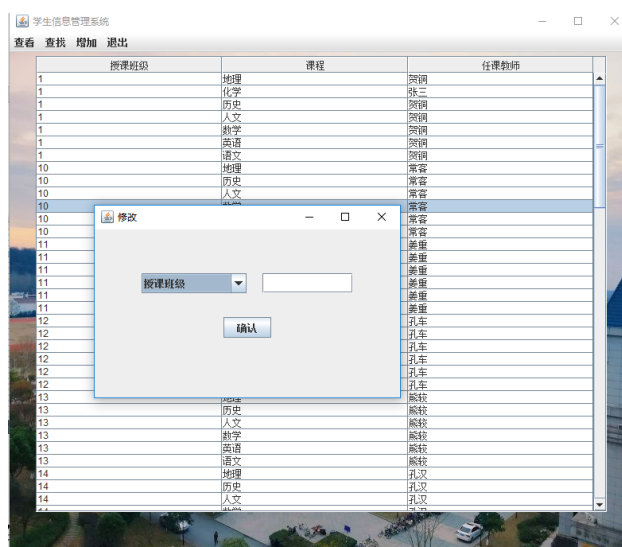
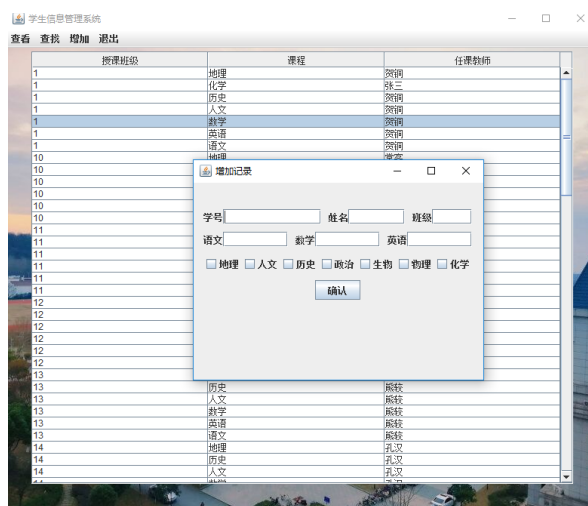
//关闭连接

```
public void closed(PreparedStatement pstmt,Connection conn) {
    if(pstmt == null) return;
    try {
        pstmt.close();
    }catch(SQLException se) {
        new ErrorPopup("Statement Closing Error!");
        se.printStackTrace();
    }
    if(conn == null) return;
    try {
        conn.close();
    }catch(SQLException se){
        new ErrorPopup("Connection Closing Error!");
        se.printStackTrace();
    }
}

public static void main(String[] args) {
    GetConnection conn = new GetConnection();
    conn.getConnection();
    System.out.println("done");
}
}
```

六、运行过程





七、 总结

总体来说，比较完整地实现了系统所需要的功能，但也暴露出来很多问题。

最重要的一个对于数据表单的设计。在学生信息管理系统中，尽管添加了一些约束，比如插入时只能插入profile中已有的学号，然而在删除过程中仍然会出现问题，导致数据冗余并且在后续操作中产生异常。由此可以发现自身对数据库理论的学习依然不够透彻，考虑仍然不够全面。这为我往后的系统设计提供了经验，也避免了很多问题。其他问题还有关于系统的构架方案。

由于缺乏设计具有用户图形界面的具有复数功能的系统的经验，在如何构架系统方面出了一些问题，尤其是用户图形界面各个控件之间的关系没有搞清楚。在最初设计用户图形界面的时候，由于各个类之间的区别并不是很明显，导致了各个类的封装性被严重破坏，产生了很多问题，对今后的代码维护和修改也带来了麻烦。这些都导致了界面代码的推翻并重构。直到使用了 MVC 设计模式，状况得到好转。由此可见设计模式也是学习的重要一环。