电 子 科 技 大 学

实 验 报 告

学生姓名：Lolipop　学 号：2018091202000　指导教师：xxx

实验地点：信软学院楼西 306　实验时间：2019-12-07/2019-12-14

一、实验室名称：信软学院楼西 306

二、实验项目名称：用 Java 面向对象方法设计并实现简易课程管理

三、实验学时：8 学时

四、实验原理：

1. 类的声明告诉编译器有一个类，但并没有为它分配空间。例如：

```
public class person;
```

2. 而类的定义即是为类加上大花括号，为类添加成员变量和成员方法等。例如：

```
public class Person {
    protected String name;
    String getName() { return this.name; }
}
```

其中，name 为类 Person 的成员变量，getName()为类 Person 的成员方法。在一个给定的源文件中，一个类只能被定义一次。如果在多个文件中定义一个类，那么每个文件中的定义必须是完全相同的。

3. 类的构造方法是与类同名的特殊方法，功能主要是完成类的初始化，例如：

```
public class Person {
    protected String name;
    Person(String readName) { this.name = readName; } //构造方法
}
```

当类实例化一个对象时会自动调用构造方法，例如：

```
public class TestPerson {
    public static void main(String[] args) {
        String name = “lolipop”;
        Person testPerson = new Person(name); //实例化对象调用构造方法
    }
}
```

4. 类的继承可以在已有类的基础上创建新类，使新创建的类自动拥有被继承类的全部成员，例如，对于前面的父类 Person：

```
public class Student extends Person{
    private String sid;
    Student(String name, String sid) {
        super(name); //继承 Person 类的 name 成员
        this.sid = sid;
    }
    String getSid() { return this.sid; }
}
```

5. 方法重载是让类以统一的方法处理不同类型数据的一种手段。在类中可以创建多个方法，他们具有相同的名字，但具有不同参数和不同的定义。调用方法时通过传递不同的参数个数与参数类型来决定具体使用哪一个方法。例如方法：

```
void readFile(Student[] students);
```

和方法：

```
void readFile(Teacher[] teachers);
```

具有不同的参数类型，故在调用方法时会自动根据给定的参数调用指定方法。

6. 方法覆盖（重写）指在子类中定义某方法与其父类有相同的名称和参数，在子类中对方法进行重新定义，需要加上@override。例如在类 Student 中的成员方法（假设父类 Person 中也有 display()方法）：

```
@Override
void display() {
    System.out.print("name: "+this.name+"\nsid: "+this.sid"\n"); //继承父类的 name
}
```

7. 对象序列化是 Java 的一种特性，被序列化的对象应调用 Serializable 接口。通过对象序列化，可以将对象转换成一系列字节，并随时能够恢复成原来的样子，实现“有限持久化”。网络传输时进行序列化，可以弥补操作系统间的差异。例如：

```
void writeFile(Student student) throws IOException, ClassNotFoundException {
    File file = new File(opfile, "studentData.dat");
    Student[] writeStudents;
    if (file.exists()) {
        FileInputStream fis = new FileInputStream(file);
        ObjectInputStream ois = new ObjectInputStream(fis);
        Student[] readStudents = (Student[]) ois.readObject();
        ois.close();
        fis.close();
        writeStudents = new Student[readStudents.length+1];
        System.arraycopy(readStudents, 0, writeStudents, 0, readStudents.length);
        writeStudents[readStudents.length] = student;
    } else {
        writeStudents = new Student[1];
        writeStudents[0] = student;
    }
    FileOutputStream fos = new FileOutputStream(file);
    ObjectOutputStream oos = new ObjectOutputStream(fos);
    oos.writeObject(writeStudents); //将对象序列化写入文件
    oos.flush();
    oos.close();
```

```
        fos.close();
}
void readFile(Student[] students) throws IOException, ClassNotFoundException {
        File file = new File(opfile, "studentData.dat");
        if (!file.exists()) {
                return;
        }
        FileInputStream fis = new FileInputStream(file);
        ObjectInputStream ois = new ObjectInputStream(fis);
        Student[] readStudents = (Student[]) ois.readObject(); //读取文件中存储的序列化对象
        System.arraycopy(readStudents, 0, students, 0, readStudents.length);
        ois.close();
}
```

文件操作时应注意异常 IOException、ClassNotFoundException 等的处理。

五、实验目的：

针对简易课程管理的开发，培养用面向对象的方法设计应用程序的能力，掌握类的声明以及对象的创建，类的成员变量和成员方法以及类的构造方法的使用；理解类的继承性，掌握方法的继承、重载和覆盖，理解类的多态性；同时，掌握对文件的操作方法，掌握 Java 流技术实现对文件的管理；掌握异常的概念以及如何定义、抛出和捕捉处理异常；培养学生在掌握 Java 语言的基本语法的基础上，编写综合应用程序的能力。

六、实验内容：

1) 定义一个名为 Person 的类，含有两个 String 类型的成员变量 name 和 sex，一个 int 类型的成员变量 age，它们用 protect 修饰符，分别实现 getXXXX 访问方法和 setXXXX 修改方法；实现构造方法 Person（String name,String sex,int age）；实现成员方法 display()显示输出类的成员变量的信息；并编写独立的测试文件测试 Person 类。
2) 定义一个名为 Student 的学生类，继承 Person 类，并且新增成员变量学号 sid 和专业 major，它们用 private 修饰符，同时对新成员变量分别编写 getXXXX

访问方法和 setXXXX 修改方法；实现构造方法 Student（String sid,String name,String sex,int age,String major）；实现成员方法 display()显示输出类的成员变量的信息；并编写独立的测试文件测试 Student 类。

3) 定义一个名为 Teacher 的教师类，继承 Person 类，并且新增成员变量职工号 tid 和职称 title，它们用 private 修饰符，同时对新成员变量分别编写 getXXXX 访问方法和 setXXXX 修改方法；实现构造方法 Teacher（String tid,String name,String sex,int age,String title）；实现成员方法 display()显示输出类的成员变量的信息；并编写独立的测试文件测试 Teacher 类。

4) 定义一个名为 Course 的课程类，含有两个 String 类型的成员变量课程名称 cname 和课程编号 cid，一个 int 类型的成员变量课时 chour，它们用 private 修饰符，分别实现 getXXXX 访问方法和 setXXXX 修改方法；实现构造方法 Course（String cid,String cname,int chour）；实现成员方法 display()显示输出类的成员变量的信息；并编写独立的测试文件测试 Course 类。

5) 定义一个名为 Schedule 的排课类，含有四个 String 类型的成员变量课程班号 classid、课程编号 cid、教师编号 tid 和上课地点 classroom，它们用 private 修饰符，分 别实现 getXXXX 访问方法和 setXXXX 修改方法；实现构造方法 Schedule（String classid,String cid,String tid,String classroom）；实现成员方法 display()显示输出类的成员变量的信息；并编写独立的测试文件测试 Schedule 类。

6) 定义一个名为 Electivecourse 的选课类，含有四个 String 类型的成员变量选课号 elid、学号 sid 和课程班号 classid，它们用 private 修饰符，分别实现 getXXXX 访问方法和 setXXXX 修改方法；实现构造方法 Electivecourse（String elid,String sid, String classid）；实现成员方法 display()显示输出类的成员变量的信息；并编写独立的测试文件测试 Electivecourse 类。

7) 定义一个名为 Myfile 的文件类，含有一个 file 类型的成员变量 opfile，分别实现 getfile（）访问方法返回 opfile、readfile(对象数组)和 Writefile(对象)；实现构造方法 myfile（文件名），用于打开已有文件，如果不存储，则新建文件；readfile(对象数组)将文件的信息读到数组中，Writefile(对象)将对象写入文件中。大家务必注意，针对前面的 6 种类的对象，分别重载 6 个 readfile

和 Writefile 成员方法。并编写独立的测试文件测试 Myfile 类。

8) 利用前面的类，编程实现分别输入 5 条记录的教师、学生、课程、排课、选课等信息，并利用文件类的方法，写入文件中；并利用自己类的 readfile 方法，将文件的信息分别读入对象数组中。编写程序任意输入学生的学号，查询显示该学生所选课程的名称、教师、上课地点。

七、实验器材（设备、元器件）：

PC 机一台，装有 Java 集成开发环境。

八、数据结构与程序：

可视化界面实验代码

代码 8-1 QuerySystemGui

```
import javax.swing.*;
import javax.swing.table.DefaultTableCellRenderer;
import javax.swing.table.DefaultTableModel;
import javax.swing.text.AttributeSet;
import javax.swing.text.PlainDocument;
import java.awt.*;
import java.awt.event.*;
import java.io.IOException;
import java.io.Serializable;
import java.util.*;

/**
 * @author Lolipop
 * @lastUpdate 2019/12/16
 */
public class QuerySystemGui {
    // 文件默认存储目录
    private static Myfile file = new Myfile(System.getProperty("user.dir"));

    // 设置最大存储数 MAXSIZE
    final private static int MAXSIZE = file.setMAXSIZE(100);

    // 状态值
    private static int NOW_CASE;
    final private static int QUERY_INFO_CASE = 1;
    final private static int STUDENT_CASE = 2;
    final private static int TEACHER_CASE = 3;
    final private static int COURSE_CASE = 4;
    final private static int SCHEDULE_CASE = 5;
    final private static int ELECTIVE_COURSE_CASE = 6;
```

```
// 常用 String 值
final private static String sysTitle = "Student Query System";
final private static String QUERY_INFO_TITLE = "查询学生选课信息";
final private static String STUDENT_TITLE = "学生信息";
final private static String TEACHER_TITLE = "教师信息";
final private static String COURSE_TITLE = "课程信息";
final private static String SCHEDULE_TITLE = "排课信息";
final private static String ELECTIVE_COURSE_TITLE = "选课信息";
final private static String ID = "序号";
final private static String NAME = "姓名";
final private static String SEX = "性别";
final private static String AGE = "年龄";
final private static String STUDENT_ID = "学号";
final private static String MAJOR = "专业";
final private static String TEACHER_ID = "教师号";
final private static String TITLE = "职称";
final private static String COURSE_ID = "课程号";
final private static String COURSE_NAME = "课程名";
final private static String COURSE_HOUR = "课时";
final private static String CLASS_ID = "班号";
final private static String CLASS_ROOM = "教室";
final private static String ELECTIVE_COURSE_ID = "选课号";
final private static String MALE = "男";
final private static String FEMALE = "女";
final private static String DIALOG_EMPTY = "信息不能为空，请检查！";
final private static String DIALOG_WRONG_INPUT = "输入有误，请检查！";

// 初始化对象数组
private static Student[] students = new Student[MAXSIZE];
private static Teacher[] teachers = new Teacher[MAXSIZE];
private static Course[] courses = new Course[MAXSIZE];
private static Schedule[] schedules = new Schedule[MAXSIZE];
private static Electivecourse[] electiveCourses = new Electivecourse[MAXSIZE];
```

```
// JFrame 框架
private static JFrame frame;

// 主面板，采用边框布局
private static JPanel mainPanel = new JPanel(new BorderLayout(5,0));

// 选项面板，采用流式布局
private static JPanel optionPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10,
0));

// 选项面板按钮
private static JButton queryInfoBtn = new JButton(QUERY_INFO_TITLE);
private static JButton studentBtn = new JButton(STUDENT_TITLE);
private static JButton teacherBtn = new JButton(TEACHER_TITLE);
private static JButton courseBtn = new JButton(COURSE_TITLE);
private static JButton scheduleBtn = new JButton(SCHEDULE_TITLE);
private static JButton electiveCourseBtn = new JButton(ELECTIVE_COURSE_TITLE);

// 表格面板，采用边框布局
private static JPanel queryInfoPanel = new JPanel(new BorderLayout());
private static JPanel studentPanel = new JPanel(new BorderLayout());
private static JPanel teacherPanel = new JPanel(new BorderLayout());
private static JPanel coursePanel = new JPanel(new BorderLayout());
private static JPanel schedulePanel = new JPanel(new BorderLayout());
private static JPanel electiveCoursePanel = new JPanel(new BorderLayout());

// 表格面板存放的表格
private static JTable queryInfoTable;
private static JTable studentTable;
private static JTable teacherTable;
private static JTable courseTable;
private static JTable scheduleTable;
private static JTable electiveCourseTable;
```

```
    private static Vector<String> queryColumnName = new Vector<>();
    private static Vector<String> studentColumnName = new Vector<>();
    private static Vector<String> teacherColumnName = new Vector<>();
    private static Vector<String> courseColumnName = new Vector<>();
    private static Vector<String> scheduleColumnName = new Vector<>();
    private static Vector<String> electiveCourseColumnName = new Vector<>();

    private static Vector<Vector<Serializable>> queryInfoRowData;
    private static Vector<Vector<Serializable>> studentRowData;
    private static Vector<Vector<Serializable>> teacherRowData;
    private static Vector<Vector<Serializable>> courseRowData;
    private static Vector<Vector<Serializable>> scheduleRowData;
    private static Vector<Vector<Serializable>> electiveCourseRowData;

    // 输入面板，采用流式布局
    private static JPanel queryInfoInputPanel = new JPanel(new
FlowLayout(FlowLayout.CENTER, 10, 15));
    private static JPanel insertStudentPanel = new JPanel(new
FlowLayout(FlowLayout.CENTER, 10, 15));
    private static JPanel insertTeacherPanel = new JPanel(new
FlowLayout(FlowLayout.CENTER, 10, 15));
    private static JPanel insertCoursePanel = new JPanel(new
FlowLayout(FlowLayout.CENTER, 10, 15));
    private static JPanel insertSchedulePanel = new JPanel(new
FlowLayout(FlowLayout.CENTER, 10, 15));
    private static JPanel insertElectiveCoursePanel = new JPanel(new
FlowLayout(FlowLayout.CENTER, 10, 15));

    // 输入面板标签
    private static JLabel studentNameLable = new JLabel(NAME);
    private static JLabel studentSexLable = new JLabel(SEX);
    private static JLabel studentAgeLable = new JLabel(AGE);
    private static JLabel teacherNameLable = new JLabel(NAME);
    private static JLabel teacherSexLable = new JLabel(SEX);
private static JLabel teacherAgeLable = new JLabel(AGE);
private static JLabel studentIdLable_insert = new JLabel(STUDENT_ID);
private static JLabel studentIdLable_query = new JLabel(STUDENT_ID);
private static JLabel studentIdLable_comboBox = new JLabel(STUDENT_ID);
private static JLabel majorLable = new JLabel(MAJOR);
private static JLabel teacherIdLable = new JLabel(TEACHER_ID);
private static JLabel teacherIdLable_comboBox = new JLabel(TEACHER_ID);
private static JLabel titleLable = new JLabel(TITLE);
private static JLabel courseIdLable = new JLabel(COURSE_ID);
private static JLabel courseIdLable_comboBox = new JLabel(COURSE_ID);
private static JLabel courseNameLable = new JLabel(COURSE_NAME);
private static JLabel courseHourLable = new JLabel(COURSE_HOUR);
private static JLabel classIdLable = new JLabel(CLASS_ID);
private static JLabel classIdLable_comboBox = new JLabel(CLASS_ID);
private static JLabel classRoomLable = new JLabel(CLASS_ROOM);
private static JLabel electiveCourseIdLable = new JLabel(ELECTIVE_COURSE_ID);

// 输入面板文本输入框
private static JTextField studentNameTextField = new JTextField(10);
private static JTextField studentAgeTextField = new JTextField(10);
private static JTextField teacherNameTextField = new JTextField(10);
private static JTextField teacherAgeTextField = new JTextField(10);
private static JTextField studentIdTextField_insert = new JTextField(10);
private static JTextField studentIdTextField_query = new JTextField(10);
private static JTextField majorTextField = new JTextField(10);
private static JTextField teacherIdTextField = new JTextField(10);
private static JTextField titleTextField = new JTextField(10);
private static JTextField courseIdTextField = new JTextField(10);
private static JTextField courseNameTextField = new JTextField(10);
private static JTextField courseHourTextField = new JTextField(10);
private static JTextField classIdTextField = new JTextField(10);
private static JTextField classRoomTextField = new JTextField(10);
private static JTextField electiveCourseIdTextField = new JTextField(10);
```

```
// 输入面板下拉框
private static JComboBox<String> studentSexComboBox = new JComboBox<>();
private static JComboBox<String> teacherSexComboBox = new JComboBox<>();
private static JComboBox<String> courseIdComboBox = new JComboBox<>();
private static JComboBox<String> teacherIdComboBox = new JComboBox<>();
private static JComboBox<String> studentIdComboBox = new JComboBox<>();
private static JComboBox<String> classIdComboBox = new JComboBox<>();

private static ArrayList<String> courseIdList = new ArrayList<>();
private static ArrayList<String> teacherIdList = new ArrayList<>();
private static ArrayList<String> studentIdList = new ArrayList<>();
private static ArrayList<String> classIdList = new ArrayList<>();

// 输入面板按钮
private static JButton queryInfoConfirm = new JButton("查询");
private static JButton insertStudentConfirm = new JButton("添加");
private static JButton insertTeacherConfirm = new JButton("添加");
private static JButton insertCourseConfirm = new JButton("添加");
private static JButton insertScheduleConfirm = new JButton("添加");
private static JButton insertElectiveCourseConfirm = new JButton("添加");
private static JButton queryInfoReset = new JButton("重置");
private static JButton insertStudentReset = new JButton("重置");
private static JButton insertTeacherReset = new JButton("重置");
private static JButton insertCourseReset = new JButton("重置");
private static JButton insertScheduleReset = new JButton("重置");
private static JButton insertElectiveCourseReset = new JButton("重置");

private static boolean insertScheduleBtn = false;
private static boolean insertElectiveCourseBtn = false;

// 提示信息窗口
private static Dialog dialog = new Dialog(frame, "提示信息", true);
private static JLabel dialogLable = new JLabel();
private static JButton dialogButton = new JButton("确认");
```

```
/*
 * Main 函数
 */
public static void main (String[] args) throws IOException, ClassNotFoundException {
        new QuerySystemGui();
}

/*
 *  初始化框架
 * @description  调用函数初始化 JFrame 框架
 */
private QuerySystemGui() throws IOException, ClassNotFoundException {
        //  设置框架属性
        frame = new JFrame(sysTitle);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(new Dimension(800, 400));
        frame.setMinimumSize(new Dimension(800, 400));
        frame.setLocationRelativeTo(null);
        // frame.setExtendedState(Frame.MAXIMIZED_BOTH);

        //  文件数据初始化
        readFile();

        //  选项面板初始化
        initOptionPanel();

        //  表格面板初始化
        initTablePanel();

        //  输入面板初始化
        initInputPanel();

        //  提示信息框初始化
        initDialogPanel();

        // 主面板初始化，启动时默认在查询选课信息界面
        mainPanel.add(optionPanel, BorderLayout.NORTH);
        mainPanel.add(queryInfoPanel, BorderLayout.CENTER);
        mainPanel.add(queryInfoInputPanel, BorderLayout.WEST);
        NOW_CASE = QUERY_INFO_CASE;

        frame.setContentPane(mainPanel);

        // 设置可见
        frame.setVisible(true);
    }

    /*
     * 读取文件
     * @description 反序列化读取文件，将获取的内容存储到对象数组中
     */
    private static void readFile() throws IOException, ClassNotFoundException {
        file.readFile(students);
        file.readFile(teachers);
        file.readFile(courses);
        file.readFile(schedules);
        file.readFile(electiveCourses);
    }

    /*
     * 初始化选项面板
     */
    private static void initOptionPanel() {
        // 监听选项面板按钮
        queryInfoBtn.addActionListener(new queryInfoBtnListener());
        studentBtn.addActionListener(new studentBtnListener());
        teacherBtn.addActionListener(new teacherBtnListener());
        courseBtn.addActionListener(new courseBtnListener());
        scheduleBtn.addActionListener(new scheduleBtnListener());
        electiveCourseBtn.addActionListener(new electiveCourseBtnListener());

        // 初始化选项面板
        optionPanel.add(queryInfoBtn);
        optionPanel.add(studentBtn);
        optionPanel.add(teacherBtn);
        optionPanel.add(courseBtn);
        optionPanel.add(scheduleBtn);
        optionPanel.add(electiveCourseBtn);
    }

    /*
     * 初始化输入面板
     */
    private static void initInputPanel() {
        // 设置输入面板大小
        Dimension inputPanelDimension = new Dimension(180, 0);
        queryInfoInputPanel.setPreferredSize(inputPanelDimension);
        insertStudentPanel.setPreferredSize(inputPanelDimension);
        insertTeacherPanel.setPreferredSize(inputPanelDimension);
        insertCoursePanel.setPreferredSize(inputPanelDimension);
        insertSchedulePanel.setPreferredSize(inputPanelDimension);
        insertElectiveCoursePanel.setPreferredSize(inputPanelDimension);

        // 设置输入面板下拉选单大小
        Dimension comboBoxDimension = new Dimension(110, 20);
        studentSexComboBox.setPreferredSize(comboBoxDimension);
        teacherSexComboBox.setPreferredSize(comboBoxDimension);
        courseIdComboBox.setPreferredSize(comboBoxDimension);
        teacherIdComboBox.setPreferredSize(comboBoxDimension);
        studentIdComboBox.setPreferredSize(comboBoxDimension);
        classIdComboBox.setPreferredSize(comboBoxDimension);
        // 监听输入面板按钮
        queryInfoConfirm.addActionListener(new queryInfoConfirmListener());
        insertStudentConfirm.addActionListener(new insertStudentConfirmListener());
        insertTeacherConfirm.addActionListener(new insertTeacherConfirmListener());
        insertCourseConfirm.addActionListener(new insertCourseConfirmListener());
        insertScheduleConfirm.addActionListener(new insertScheduleConfirmListener());
        insertElectiveCourseConfirm.addActionListener(new
insertElectiveCourseConfirmListener());

        queryInfoReset.addActionListener(new queryInfoResetListener());
        insertStudentReset.addActionListener(new insertStudentResetListener());
        insertTeacherReset.addActionListener(new insertTeacherResetListener());
        insertCourseReset.addActionListener(new insertCourseResetListener());
        insertScheduleReset.addActionListener(new insertScheduleResetListener());
        insertElectiveCourseReset.addActionListener(new
insertElectiveCourseResetListener());

        // 添加输入面板下拉选单数据
        studentSexComboBox.addItem(MALE);
        studentSexComboBox.addItem(FEMALE);
        teacherSexComboBox.addItem(MALE);
        teacherSexComboBox.addItem(FEMALE);
        getComboBox();

        // 如果排课信息或选课信息下拉选单不存在数据则隐藏其按钮
        insertScheduleBtn = checkInsertScheduleComboBox();
        insertElectiveCourseBtn = checkInsertElectiveCourseComboBox();

        // 年龄、课时信息只允许输入数字
        studentAgeTextField.setDocument(new integerTextField());
        teacherAgeTextField.setDocument(new integerTextField());
        courseHourTextField.setDocument(new integerTextField());
```

```
// 初始化查询界面输入面板
queryInfoInputPanel.add(studentIdLable_query);
queryInfoInputPanel.add(studentIdTextField_query);
queryInfoInputPanel.add(queryInfoConfirm);
queryInfoInputPanel.add(queryInfoReset);

// 初始化学生信息界面输入面板
insertStudentPanel.add(studentIdLable_insert);
insertStudentPanel.add(studentIdTextField_insert);
insertStudentPanel.add(studentNameLable);
insertStudentPanel.add(studentNameTextField);
insertStudentPanel.add(studentSexLable);
insertStudentPanel.add(studentSexComboBox);
insertStudentPanel.add(studentAgeLable);
insertStudentPanel.add(studentAgeTextField);
insertStudentPanel.add(majorLable);
insertStudentPanel.add(majorTextField);
insertStudentPanel.add(insertStudentConfirm);
insertStudentPanel.add(insertStudentReset);

// 初始化教师信息界面输入面板
insertTeacherPanel.add(teacherIdLable);
insertTeacherPanel.add(teacherIdTextField);
insertTeacherPanel.add(teacherNameLable);
insertTeacherPanel.add(teacherNameTextField);
insertTeacherPanel.add(teacherSexLable);
insertTeacherPanel.add(teacherSexComboBox);
insertTeacherPanel.add(teacherAgeLable);
insertTeacherPanel.add(teacherAgeTextField);
insertTeacherPanel.add(titleLable);
insertTeacherPanel.add(titleTextField);
insertTeacherPanel.add(insertTeacherConfirm);
insertTeacherPanel.add(insertTeacherReset);
```

```
        // 初始化课程信息界面输入面板
        insertCoursePanel.add(courseIdLable);
        insertCoursePanel.add(courseIdTextField);
        insertCoursePanel.add(courseNameLable);
        insertCoursePanel.add(courseNameTextField);
        insertCoursePanel.add(courseHourLable);
        insertCoursePanel.add(courseHourTextField);
        insertCoursePanel.add(insertCourseConfirm);
        insertCoursePanel.add(insertCourseReset);

        // 初始化排课信息界面输入面板
        insertSchedulePanel.add(classIdLable);
        insertSchedulePanel.add(classIdTextField);
        insertSchedulePanel.add(courseIdLable_comboBox);
        insertSchedulePanel.add(courseIdComboBox);
        insertSchedulePanel.add(teacherIdLable_comboBox);
        insertSchedulePanel.add(teacherIdComboBox);
        insertSchedulePanel.add(classRoomLable);
        insertSchedulePanel.add(classRoomTextField);
        insertSchedulePanel.add(insertScheduleConfirm);
        insertSchedulePanel.add(insertScheduleReset);

        // 初始化选课信息界面输入面板
        insertElectiveCoursePanel.add(electiveCourseIdLable);
        insertElectiveCoursePanel.add(electiveCourseIdTextField);
        insertElectiveCoursePanel.add(studentIdLable_comboBox);
        insertElectiveCoursePanel.add(studentIdComboBox);
        insertElectiveCoursePanel.add(classIdLable_comboBox);
        insertElectiveCoursePanel.add(classIdComboBox);
        insertElectiveCoursePanel.add(insertElectiveCourseConfirm);
        insertElectiveCoursePanel.add(insertElectiveCourseReset);
    }

    /*
```

```
 *  初始化表格面板
 */
private static void initTablePanel() {
    //  初始化表格列标题
    queryColumnName.add(ID);
    queryColumnName.add(COURSE_NAME);
    queryColumnName.add(NAME);
    queryColumnName.add(CLASS_ROOM);
    studentColumnName.add(ID);
    studentColumnName.add(STUDENT_ID);
    studentColumnName.add(NAME);
    studentColumnName.add(SEX);
    studentColumnName.add(AGE);
    studentColumnName.add(MAJOR);
    teacherColumnName.add(ID);
    teacherColumnName.add(TEACHER_ID);
    teacherColumnName.add(NAME);
    teacherColumnName.add(SEX);
    teacherColumnName.add(AGE);
    teacherColumnName.add(TITLE);
    courseColumnName.add(ID);
    courseColumnName.add(COURSE_ID);
    courseColumnName.add(COURSE_NAME);
    courseColumnName.add(COURSE_HOUR);
    scheduleColumnName.add(ID);
    scheduleColumnName.add(CLASS_ID);
    scheduleColumnName.add(COURSE_ID);
    scheduleColumnName.add(TEACHER_ID);
    scheduleColumnName.add(CLASS_ROOM);
    electiveCourseColumnName.add(ID);
    electiveCourseColumnName.add(ELECTIVE_COURSE_ID);
    electiveCourseColumnName.add(STUDENT_ID);
    electiveCourseColumnName.add(CLASS_ID);
```

```
        // 初始化表格行数据
        queryInfoRowData = getRowData();
        studentRowData = getRowData(students);
        teacherRowData = getRowData(teachers);
        courseRowData = getRowData(courses);
        scheduleRowData = getRowData(schedules);
        electiveCourseRowData = getRowData(electiveCourses);

        // 初始化表格值
        queryInfoTable = new JTable(queryInfoRowData, queryColumnName);
        studentTable = new JTable(studentRowData, studentColumnName);
        teacherTable = new JTable(teacherRowData, teacherColumnName);
        courseTable = new JTable(courseRowData, courseColumnName);
        scheduleTable = new JTable(scheduleRowData, scheduleColumnName);
        electiveCourseTable = new JTable(electiveCourseRowData,
electiveCourseColumnName);

        // 设置表格表头格式
        // 表头字体
        Font tableHeaderFont = new Font("SimHei", Font.BOLD, 16);
        queryInfoTable.getTableHeader().setFont(tableHeaderFont);
        studentTable.getTableHeader().setFont(tableHeaderFont);
        teacherTable.getTableHeader().setFont(tableHeaderFont);
        courseTable.getTableHeader().setFont(tableHeaderFont);
        scheduleTable.getTableHeader().setFont(tableHeaderFont);
        electiveCourseTable.getTableHeader().setFont(tableHeaderFont);

        // 设置表格内容格式
        // 内容字体
        Font rowDataFont = new Font("SimHei", Font.PLAIN, 14);
        queryInfoTable.setFont(rowDataFont);
        studentTable.setFont(rowDataFont);
        teacherTable.setFont(rowDataFont);
        courseTable.setFont(rowDataFont);
```

```
scheduleTable.setFont(rowDataFont);
electiveCourseTable.setFont(rowDataFont);

// 内容行高
queryInfoTable.setRowHeight(22);
studentTable.setRowHeight(22);
teacherTable.setRowHeight(22);
courseTable.setRowHeight(22);
scheduleTable.setRowHeight(22);
electiveCourseTable.setRowHeight(22);

// 内容居中
DefaultTableCellRenderer cr = new DefaultTableCellRenderer();
cr.setHorizontalAlignment(JLabel.CENTER);
queryInfoTable.setDefaultRenderer(Object.class, cr);
studentTable.setDefaultRenderer(Object.class, cr);
teacherTable.setDefaultRenderer(Object.class, cr);
courseTable.setDefaultRenderer(Object.class, cr);
scheduleTable.setDefaultRenderer(Object.class, cr);
electiveCourseTable.setDefaultRenderer(Object.class, cr);

// 初始化表格面板并添加滚动条
queryInfoPanel.add(queryInfoTable, BorderLayout.CENTER);
queryInfoPanel.add(new JScrollPane(queryInfoTable));
studentPanel.add(studentTable, BorderLayout.CENTER);
studentPanel.add(new JScrollPane(studentTable));
teacherPanel.add(teacherTable, BorderLayout.CENTER);
teacherPanel.add(new JScrollPane(teacherTable));
coursePanel.add(courseTable, BorderLayout.CENTER);
coursePanel.add(new JScrollPane(courseTable));
schedulePanel.add(scheduleTable, BorderLayout.CENTER);
schedulePanel.add(new JScrollPane(scheduleTable));
electiveCoursePanel.add(electiveCourseTable, BorderLayout.CENTER);
electiveCoursePanel.add(new JScrollPane(electiveCourseTable));
```

```
}

/*
 *  获取表格信息
 * @description  读取文件中获取的对象数组的内容并返回存放表格数据的对象数组
 */
private static Vector<Vector<java.io.Serializable>> getRowData() {
    //  返回未存放数据的 Vector
    return new Vector<>();
}

private static Vector<Vector<java.io.Serializable>> getRowData(Student[] students) {
    int count = 0;
    Vector<Vector<java.io.Serializable>> rowData = new Vector<>();
    for (Student student : students) {
        if (student != null) {
            Vector<java.io.Serializable> line = new Vector<>();
            line.add(count+1);
            line.add(student.getSid());
            line.add(student.getName());
            line.add(student.getSex());
            line.add(student.getAge());
            line.add(student.getMajor());
            rowData.add(line);
            count++;
        }
        else break;
    }
    return rowData;
}

private static Vector<Vector<java.io.Serializable>> getRowData(Teacher[] teachers) {
    int count = 0;
    Vector<Vector<java.io.Serializable>> rowData = new Vector<>();
```

```
        for (Teacher teacher : teachers) {
            if (teacher != null) {
                Vector<java.io.Serializable> line = new Vector<>();
                line.add(count+1);
                line.add(teacher.getTid());
                line.add(teacher.getName());
                line.add(teacher.getSex());
                line.add(teacher.getAge());
                line.add(teacher.getTitle());
                rowData.add(line);
                count++;
            }
            else break;
        }
        return rowData;
    }

    private static Vector<Vector<java.io.Serializable>> getRowData(Course[] courses) {
        int count = 0;
        Vector<Vector<java.io.Serializable>> rowData = new Vector<>();
        for (Course course : courses) {
            if (course != null) {
                Vector<java.io.Serializable> line = new Vector<>();
                line.add(count+1);
                line.add(course.getCid());
                line.add(course.getCname());
                line.add(course.getChour());
                rowData.add(line);
                count++;
            }
            else break;
        }
        return rowData;
    }
    private static Vector<Vector<java.io.Serializable>> getRowData(Schedule[] schedules) {
        int count = 0;
        Vector<Vector<java.io.Serializable>> rowData = new Vector<>();
        for (Schedule schedule : schedules) {
            if (schedule != null) {
                Vector<java.io.Serializable> line = new Vector<>();
                line.add(count+1);
                line.add(schedule.getClassid());
                line.add(schedule.getCid());
                line.add(schedule.getTid());
                line.add(schedule.getClassroom());
                rowData.add(line);
                count++;
            }
            else break;
        }
        return rowData;
    }

    private static Vector<Vector<java.io.Serializable>> getRowData(Electivecourse[]
electiveCourses) {
        int count = 0;
        Vector<Vector<java.io.Serializable>> rowData = new Vector<>();
        for (Electivecourse electivecourse : electiveCourses) {
            if (electivecourse != null) {
                Vector<java.io.Serializable> line = new Vector<>();
                line.add(count+1);
                line.add(electivecourse.getElid());
                line.add(electivecourse.getSid());
                line.add(electivecourse.getClassid());
                rowData.add(line);
                count++;
            }
            else break;
        }
        return rowData;
    }

    /*
     *  初始化提示信息窗口
     * @description  生成一个具有提示信息的新窗口
     */
    private static void initDialogPanel() {
        dialog.setLayout(new FlowLayout(FlowLayout.CENTER));

        dialogButton.addActionListener(e-> dialog.setVisible(false));
        dialog.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                super.windowClosing(e);
                dialog.setVisible(false);
            }
        });

        dialog.add(dialogLable);
        dialog.add(dialogButton);
    }

    /*
     *  显示提示信息窗口
     * @description  修改提示信息并使提示窗口可见
     */
    private static void dialog(String words) {
        dialog.setSize(new Dimension(200, 100));
        dialog.setLocationRelativeTo(null);

        dialogLable.setText(words);
        dialog.setVisible(true);
}

/*
 *  选项按钮监听器
 * @description  点击时切换工作面板
 */
private static class queryInfoBtnListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
                if (NOW_CASE != QUERY_INFO_CASE) {
                        mainPanel.removeAll();
                        mainPanel.add(optionPanel, BorderLayout.NORTH);
                        mainPanel.add(queryInfoPanel, BorderLayout.CENTER);
                        mainPanel.add(queryInfoInputPanel, BorderLayout.WEST);
                        mainPanel.updateUI();
                        mainPanel.repaint();

                        NOW_CASE = QUERY_INFO_CASE;
                        frame.setTitle(sysTitle+" - "+QUERY_INFO_TITLE);
                }
        }
}

private static class studentBtnListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
                if (NOW_CASE != STUDENT_CASE) {
                        mainPanel.removeAll();
                        mainPanel.add(optionPanel, BorderLayout.NORTH);
                        mainPanel.add(studentPanel, BorderLayout.CENTER);
                        mainPanel.add(insertStudentPanel, BorderLayout.WEST);
                        mainPanel.updateUI();
                        mainPanel.repaint();
                NOW_CASE = STUDENT_CASE;
                frame.setTitle(sysTitle+" - "+STUDENT_TITLE);
            }
        }
    }

    private static class teacherBtnListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (NOW_CASE != TEACHER_CASE) {
                mainPanel.removeAll();
                mainPanel.add(optionPanel, BorderLayout.NORTH);
                mainPanel.add(teacherPanel, BorderLayout.CENTER);
                mainPanel.add(insertTeacherPanel, BorderLayout.WEST);
                mainPanel.updateUI();
                mainPanel.repaint();

                NOW_CASE = TEACHER_CASE;
                frame.setTitle(sysTitle+" - "+TEACHER_TITLE);
            }
        }
    }

    private static class courseBtnListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (NOW_CASE != COURSE_CASE) {
                mainPanel.removeAll();
                mainPanel.add(optionPanel, BorderLayout.NORTH);
                mainPanel.add(coursePanel, BorderLayout.CENTER);
                mainPanel.add(insertCoursePanel, BorderLayout.WEST);
                mainPanel.updateUI();
                mainPanel.repaint();
                NOW_CASE = COURSE_CASE;
                frame.setTitle(sysTitle+" - "+COURSE_TITLE);
            }
        }
    }

    private static class scheduleBtnListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (NOW_CASE != SCHEDULE_CASE) {
                mainPanel.removeAll();
                mainPanel.add(optionPanel, BorderLayout.NORTH);
                mainPanel.add(schedulePanel, BorderLayout.CENTER);
                mainPanel.add(insertSchedulePanel, BorderLayout.WEST);
                mainPanel.updateUI();
                mainPanel.repaint();

                NOW_CASE = STUDENT_CASE;
                frame.setTitle(sysTitle+" - "+SCHEDULE_TITLE);

                // 如果排课信息输入面板按钮被隐藏，则重新进行判断
                if (!insertScheduleBtn) {
                    insertScheduleBtn = checkInsertScheduleComboBox();
                }
            }
        }
    }

    private static class electiveCourseBtnListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (NOW_CASE != ELECTIVE_COURSE_CASE) {
                mainPanel.removeAll();
                mainPanel.add(optionPanel, BorderLayout.NORTH);
                mainPanel.add(electiveCoursePanel, BorderLayout.CENTER);
                mainPanel.add(insertElectiveCoursePanel, BorderLayout.WEST);
                mainPanel.updateUI();
                mainPanel.repaint();

                NOW_CASE = ELECTIVE_COURSE_CASE;
                frame.setTitle(sysTitle+" - "+ELECTIVE_COURSE_TITLE);

                if (!insertElectiveCourseBtn) {
                    insertElectiveCourseBtn = checkInsertElectiveCourseComboBox();
                }
            }
        }
    }

    /*
     * 输入面板确定按钮监听器
     * @description 点击时检索信息或将数据写入文件，并刷新表单
     */
    private static class queryInfoConfirmListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            String studentId = studentIdTextField_query.getText();

            // 检查输入是否为空
            if (studentId.equals("")) {
                dialog(STUDENT_ID+DIALOG_EMPTY);
                return;
            }

            boolean hasElectiveCourse = false;
            int count = 0;
            String classroom;
```

```
String courseName = null;
String teacherName = null;

// 根据学号检索选课类，获得班级号
for (Electivecourse electivecourse : electiveCourses) {
    if (electivecourse != null && studentId.equals(electivecourse.getSid())) {
        String classId = electivecourse.getClassid();

        // 根据班级号检索排课类，获得课程号、教师号和上课教室
        for (Schedule schedule : schedules) {
            if (schedule != null && classId.equals(schedule.getClassid())) {
                String courseId = schedule.getCid();
                String teacherId = schedule.getTid();
                classroom = schedule.getClassroom();

                // 根据课程号、教师号分别获取课程名称和教师名称
                for (Course course : courses) {
                    if (course != null && courseId.equals(course.getCid())) {
                        courseName = course.getCname();
                        break;
                    }
                }

                for (Teacher teacher : teachers) {
                    if (teacher != null && teacherId.equals(teacher.getTid()))
{
                        teacherName = teacher.getName();
                        break;
                    }
                }

                // 查询的结果
                count++;
                Vector<java.io.Serializable> line = new Vector<>();
```

```
                            line.add(count);
                            line.add(courseName);
                            line.add(teacherName);
                            line.add(classroom);
                            queryInfoRowData.add(line);

                            // 设置布尔值为真
                            hasElectiveCourse = true;
                            break;
                        }
                    }
                }
            }

            // 如果为查询到信息，则显示 dialog
            if (!hasElectiveCourse) {
                dialog("未查询到"+studentId+"的选课信息");
            }
            else    queryInfoTable.setModel(new    DefaultTableModel(queryInfoRowData,
queryColumnName));

            // 修改框架名
            frame.setTitle(sysTitle+" - "+QUERY_INFO_TITLE+" - "+studentId);

            // 点击查询按钮后选中输入的学号
            studentIdTextField_query.requestFocus();
            studentIdTextField_query.selectAll();
        }
    }

    private static class insertStudentConfirmListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (studentAgeTextField.getText().equals("")) {
```

```
        dialog(DIALOG_EMPTY);
        return;
    }

    int studentAge = Integer.parseInt(studentAgeTextField.getText());

    // 检查输入值
    if (checkAge(studentAge)) {
        dialog("学生"+AGE+DIALOG_WRONG_INPUT);
        studentAgeTextField.requestFocus();
        studentAgeTextField.selectAll();
        return;
    }

    String studentSex = Objects.requireNonNull(studentSexComboBox.getSelectedItem()).toString();
    String studentId = studentIdTextField_insert.getText();
    String studentName = studentNameTextField.getText();
    String major = majorTextField.getText();

    if (studentSex.equals("") || studentId.equals("")
            || studentName.equals("") || major.equals("")) {
        dialog(DIALOG_EMPTY);
        return;
    }

    // 将输入内容写入文件并重新读取文件到对象数组 students 中
    try {
        file.writeFile(new Student(studentName, studentSex, studentAge, studentId, major));
        file.readFile(students);
    } catch (IOException | ClassNotFoundException ex) {
        ex.printStackTrace();
    }
```

```
            // 刷新表格
            Vector<java.io.Serializable> line = new Vector<>();
            line.add(studentTable.getRowCount()+1);
            line.add(studentId);
            line.add(studentName);
            line.add(studentSex);
            line.add(studentAge);
            line.add(major);
            studentRowData.add(line);
            studentTable.setModel(new DefaultTableModel(studentRowData,
studentColumnName));

            // 重置信息
            studentIdTextField_insert.setText("");
            studentNameTextField.setText("");

            studentIdTextField_insert.requestFocus();

            // 刷新下拉选单
            studentIdComboBox.addItem(studentId);
        }
    }

    private static class insertTeacherConfirmListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            if (teacherAgeTextField.getText().equals("")) {
                dialog(DIALOG_EMPTY);
                return;
            }

            int teacherAge = Integer.parseInt(teacherAgeTextField.getText());
            if (checkAge(teacherAge)) {
                dialog("教师"+AGE+DIALOG_WRONG_INPUT);
                teacherAgeTextField.requestFocus();
                teacherAgeTextField.selectAll();
                return;
            }

            String teacherSex = Objects.requireNonNull(teacherSexComboBox.getSelectedItem()).toString();
            String teacherId = teacherIdTextField.getText();
            String teacherName = teacherNameTextField.getText();
            String title = titleTextField.getText();

            if (teacherSex.equals("") || teacherId.equals("") || teacherName.equals("") || title.equals("")) {
                dialog(DIALOG_EMPTY);
                return;
            }

            try {
                file.writeFile(new Teacher(teacherName, teacherSex, teacherAge, teacherId, title));
                file.readFile(teachers);
            } catch (IOException | ClassNotFoundException ex) {
                ex.printStackTrace();
            }

            Vector<java.io.Serializable> line = new Vector<>();
            line.add(teacherTable.getRowCount()+1);
            line.add(teacherId);
            line.add(teacherName);
            line.add(teacherSex);
            line.add(teacherAge);
            line.add(title);
```

```
                teacherRowData.add(line);
                teacherTable.setModel(new                    DefaultTableModel(teacherRowData,
teacherColumnName));

                teacherIdTextField.setText("");
                teacherNameTextField.setText("");
                teacherAgeTextField.setText("");

                teacherIdTextField.requestFocus();

                teacherIdComboBox.addItem(teacherId);
            }
        }

        private static class insertCourseConfirmListener implements ActionListener {
            @Override
            public void actionPerformed(ActionEvent e) {
                if (courseHourTextField.getText().equals("")) {
                    dialog(DIALOG_EMPTY);
                    return;
                }

                int courseHour = Integer.parseInt(courseHourTextField.getText());

                if (checkCourseHour(courseHour)) {
                    dialog(COURSE_HOUR+DIALOG_WRONG_INPUT);
                    courseHourTextField.requestFocus();
                    courseHourTextField.selectAll();
                    return;
                }

                String courseId = courseIdTextField.getText();
                String courseName = courseNameTextField.getText();
                if (courseId.equals("") || courseName.equals("")) {
                    dialog(DIALOG_EMPTY);
                    return;
                }

                try {
                    file.writeFile(new Course(courseName, courseId, courseHour));
                    file.readFile(courses);
                } catch (IOException | ClassNotFoundException ex) {
                    ex.printStackTrace();
                }

                Vector<java.io.Serializable> line = new Vector<>();
                line.add(courseTable.getRowCount()+1);
                line.add(courseId);
                line.add(courseName);
                line.add(courseHour);
                courseRowData.add(line);
                courseTable.setModel(new DefaultTableModel(courseRowData,
courseColumnName));

                courseIdTextField.setText("");
                courseNameTextField.setText("");
                courseHourTextField.setText(" ");

                courseIdTextField.requestFocus();

                courseIdComboBox.addItem(courseId);
            }
        }

        private static class insertScheduleConfirmListener implements ActionListener {
            @Override
            public void actionPerformed(ActionEvent e) {
```

```
        String classId = classIdTextField.getText();
        String courseId = Objects.requireNonNull(courseIdComboBox.getSelectedItem()).toString();
        String teacherId = Objects.requireNonNull(teacherIdComboBox.getSelectedItem()).toString();
        String classRoom = classRoomTextField.getText();

        if (classId.equals("") || courseId.equals("") || teacherId.equals("") || classRoom.equals("")) {
            dialog(DIALOG_EMPTY);
            return;
        }

        try {
            file.writeFile(new Schedule(classId, courseId, teacherId, classRoom));
            file.readFile(schedules);
        } catch (IOException | ClassNotFoundException ex) {
            ex.printStackTrace();
        }

        Vector<java.io.Serializable> line = new Vector<>();
        line.add(scheduleTable.getRowCount()+1);
        line.add(classId);
        line.add(courseId);
        line.add(teacherId);
        line.add(classRoom);
        scheduleRowData.add(line);
        scheduleTable.setModel(new DefaultTableModel(scheduleRowData, scheduleColumnName));

        classIdTextField.setText("");
        courseIdComboBox.setSelectedIndex(0);
        classRoomTextField.setText("");
```

```
                classIdTextField.requestFocus();

                classIdComboBox.addItem(classId);
            }
        }

        private static class insertElectiveCourseConfirmListener implements ActionListener {
            @Override
            public void actionPerformed(ActionEvent e) {
                String electiveCourseId = electiveCourseIdTextField.getText();
                String                                   studentId                                    =
Objects.requireNonNull(studentIdComboBox.getSelectedItem()).toString();
                String                                   classId                                      =
Objects.requireNonNull(classIdComboBox.getSelectedItem()).toString();

                if (electiveCourseId.equals("") || studentId.equals("") || classId.equals("")) {
                    dialog(DIALOG_EMPTY);
                    return;
                }

                try {
                    file.writeFile(new Electivecourse(classId, electiveCourseId, studentId));
                    file.readFile(electiveCourses);
                } catch (IOException | ClassNotFoundException ex) {
                    ex.printStackTrace();
                }

                Vector<java.io.Serializable> line = new Vector<>();
                line.add(electiveCourseTable.getRowCount()+1);
                line.add(electiveCourseId);
                line.add(studentId);
                line.add(classId);
                electiveCourseRowData.add(line);
                electiveCourseTable.setModel(new     DefaultTableModel(electiveCourseRowData,
electiveCourseColumnName));

                electiveCourseIdTextField.setText("");
                classIdComboBox.setSelectedIndex(0);

                electiveCourseIdTextField.requestFocus();
            }
        }

        /*
         *  输入面板重置按钮监听器
         * @description  点击时将输入面板清零
         */
        private static class queryInfoResetListener implements ActionListener {
            @Override
            public void actionPerformed(ActionEvent e) {
                //  清除输入学号文本框并选中文本框
                studentIdTextField_query.setText("");
                studentIdTextField_query.requestFocus();
            }
        }

        private static class insertStudentResetListener implements ActionListener {
            @Override
            public void actionPerformed(ActionEvent e) {
                studentIdTextField_insert.setText("");
                studentNameTextField.setText("");
                studentSexComboBox.setSelectedIndex(0);
                studentAgeTextField.setText("");
                majorTextField.setText("");

                studentIdTextField_insert.requestFocus();
            }
        }
```

```
private static class insertTeacherResetListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
                teacherIdTextField.setText("");
                teacherNameTextField.setText("");
                teacherSexComboBox.setSelectedIndex(0);
                teacherAgeTextField.setText("");
                titleTextField.setText("");

                teacherIdTextField.requestFocus();
        }
}

private static class insertCourseResetListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
                courseIdTextField.setText("");
                courseNameTextField.setText("");
                courseHourTextField.setText("");

                courseIdTextField.requestFocus();
        }
}

private static class insertScheduleResetListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
                classIdTextField.setText("");
                courseIdComboBox.setSelectedIndex(0);
                teacherIdComboBox.setSelectedIndex(0);
                classRoomTextField.setText("");

                classIdTextField.requestFocus();
        }
    }

    private static class insertElectiveCourseResetListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent e) {
            electiveCourseIdTextField.setText("");
            studentIdComboBox.setSelectedIndex(0);
            classIdComboBox.setSelectedIndex(0);

            electiveCourseIdTextField.requestFocus();
        }
    }

    /*
     *  数字输入检验
     * @description  限制年龄、课时输入只能输入数字
     */
    private static class integerTextField extends PlainDocument {
        public integerTextField() {
            super();
        }

        public void insertString(int offset, String str, AttributeSet attr) throws
javax.swing.text.BadLocationException {
            if (str == null) {
                return;
            }

            char[] s = str.toCharArray();
            int length = 0;
            //  过滤非数字
            for (int i = 0; i < s.length; i++) {
                if ((s[i] >= '0') && (s[i] <= '9')) {
                    s[length++] = s[i];
                }
                super.insertString(offset, new String(s, 0, length), attr);
            }
        }
    }

    /*
     * 数据输入检验
     * @description 检验输入的年龄、课时数据是否合理
     */
    private static boolean checkAge(int age) {
        // 当输入的年龄大于 120 或小于 0 时，输入数据无效
        return (age < 0 || age > 120);
    }

    private static boolean checkCourseHour(int courseHour) {
        // 当输入的课时大于 240 或小于 0 时，输入数据无效
        return (courseHour < 0 || courseHour > 240);
    }

    /*
     * 获取输入面板下拉选单
     * @description 将文件读取到的信息添加到下拉选单中
     */
    private static void getComboBox() {
        // 遍历信息添加到 ArrayList 中
        for (Course course : courses) {
            if (course != null)
                courseIdList.add(course.getCid());
            else break;
        }
        for (Teacher teacher : teachers) {
            if (teacher != null)
```

```
                teacherIdList.add(teacher.getTid());
            else break;
        }
        for (Student student : students) {
            if (student != null)
                studentIdList.add(student.getSid());
            else break;
        }
        for (Schedule schedule : schedules) {
            if (schedule != null)
                classIdList.add(schedule.getClassid());
            else break;
        }

        // 对信息进行排序
        Collections.sort(courseIdList);
        Collections.sort(teacherIdList);
        Collections.sort(studentIdList);
        Collections.sort(classIdList);

        // 将排序后的信息添加到下拉选单中
        for (String courseid : courseIdList) {
            courseIdComboBox.addItem(courseid);
        }
        for (String teacherid : teacherIdList) {
            teacherIdComboBox.addItem(teacherid);
        }
        for (String studentid : studentIdList) {
            studentIdComboBox.addItem(studentid);
        }
        for (String classid : classIdList) {
            classIdComboBox.addItem(classid);
        }
    }
    /*
     * 检验输入面板下拉选单
     * @description 如果不存在则设置对应面板按钮不可见
     */
    private static boolean checkInsertElectiveCourseComboBox() {
        if (studentIdComboBox.getSelectedItem() == null ||
classIdComboBox.getSelectedItem() == null) {
            insertElectiveCourseConfirm.setVisible(false);
            insertElectiveCourseReset.setVisible(false);
            return false;
        }
        else {
            insertElectiveCourseConfirm.setVisible(true);
            insertElectiveCourseReset.setVisible(true);
            return true;
        }
    }
    private static boolean checkInsertScheduleComboBox() {
        if (courseIdComboBox.getSelectedItem() == null ||
teacherIdComboBox.getSelectedItem() == null) {
            insertScheduleConfirm.setVisible(false);
            insertScheduleReset.setVisible(false);
            return false;
        }
        else {
            insertScheduleConfirm.setVisible(true);
            insertScheduleReset.setVisible(true);
            return true;
        }
    }
}
```

终端界面实验代码

代码 8-2 QuerySystem

```
import java.io.IOException;
import java.util.Scanner;

/**
  * @author Lolipop
  * @lastUpdate 2019/12/14
  */
public class QuerySystem {
        private static Scanner scanner = new Scanner(System.in);

        // 文件默认存储在工作目录
        private static Myfile file = new Myfile(System.getProperty("user.dir"));

        // 设置最大存储数 MAXSIZE
        private static int MAXSIZE = file.setMAXSIZE(30);

        // 初始化变量
        private static String name;
        private static String sex;
        private static int age;
        private static String sid;
        private static String major;
        private static String tid;
        private static String title;
        private static String cname;
        private static String cid;
        private static int chour;
        private static String classid;
        private static String classroom;
        private static String elid;

        // 初始化对象数组
```

```
private static Student[] students = new Student[MAXSIZE];
private static Teacher[] teachers = new Teacher[MAXSIZE];
private static Course[] courses = new Course[MAXSIZE];
private static Schedule[] schedules = new Schedule[MAXSIZE];
private static Electivecourse[] electivecourses = new Electivecourse[MAXSIZE];

// choice 选项值
final private static int QUIT_CASE = 0;
final private static int QUERY_INFO_CASE = 1;
final private static int INSERT_STUDENT_CASE = 2;
final private static int INSERT_TEACHER_CASE = 3;
final private static int INSERT_COURSE_CASE = 4;
final private static int INSERT_SCHEDULE_CASE = 5;
final private static int INSERT_ELECTIVECOURSE_CASE = 6;

public static void main(String[] args) throws IOException, ClassNotFoundException {
    int choice;
    boolean flag = true;
    readFile();

    while(flag) {
        choice = querySys();
        switch (choice) {
            case QUIT_CASE:
                System.out.println("You quit successfully!");
                flag = false;
                break;
            case QUERY_INFO_CASE:
                System.out.println("---Query student information---");
                queryInfo();
                break;
            case INSERT_STUDENT_CASE:
                System.out.println("---Insert new student information---");
                insertStudent();
```

```
                    break;
                case INSERT_TEACHER_CASE:
                    System.out.println("---Insert new teacher information---");
                    insertTeacher();
                    break;
                case INSERT_COURSE_CASE:
                    System.out.println("---Insert new course information---");
                    insertCourse();
                    break;
                case INSERT_SCHEDULE_CASE:
                    System.out.println("---Insert new schedule information---");
                    insertSchedule();
                    break;
                case INSERT_ELECTIVECOURSE_CASE:
                    System.out.println("---Insert new elective course information---");
                    insertElectivecourse();
                    break;
                default: System.out.println("Wrong code! Check your input."); break;
            }
        }
    }

    // 显示操作提示界面并获取选项值
    private static int querySys() {
        System.out.println("\n---------Query Information System---------");
        System.out.println("#"+QUERY_INFO_CASE+"    Query student information");
        System.out.println("#"+INSERT_STUDENT_CASE+"    Insert new student information");
        System.out.println("#"+INSERT_TEACHER_CASE+"    Insert new teacher information");
        System.out.println("#"+INSERT_COURSE_CASE+"    Insert new course information");
        System.out.println("#"+INSERT_SCHEDULE_CASE+"    Insert new schedule information");
        System.out.println("#"+INSERT_ELECTIVECOURSE_CASE+"    Insert new elective course information");
        System.out.println("#"+QUIT_CASE+"    Quit system");
        System.out.println("----------------------------------------");
        System.out.print("input choice: #");
        return scanner.nextInt();
    }

    private static void queryInfo() {
        System.out.println("input student id your want to query below:");
        sid = scanner.next();

        // 根据学号检索选课类，获得班级号
        for (Electivecourse electivecourse : electivecourses) {
            if (electivecourse != null && sid.equals(electivecourse.getSid())) {
                classid = electivecourse.getClassid();

                // 根据班级号检索排课类，获得课程号、教师号和上课教室
                for (Schedule schedule : schedules) {
                    if (schedule != null && classid.equals(schedule.getClassid())) {
                        cid = schedule.getCid();
                        tid = schedule.getTid();
                        classroom = schedule.getClassroom();

                        // 根据课程号、教师号分别获取课程名称和教师名称
                        for (Course course : courses) {
                            if (course != null && cid.equals(course.getCid())) {
                                cname = course.getCname();
                                break;
                            }
                        }

                        for (Teacher teacher : teachers) {
                            if (teacher != null && tid.equals(teacher.getTid())) {
                                name = teacher.getName();
                                break;
                            }
                        }

                        // 输出显示查询的结果
                        outputQueryInfo(cname, name, classroom);
                        break;
                    }
                }
            }
        }
    }

    private static void outputQueryInfo(String courseName, String teacherName, String
classroom) {
        System.out.println("---Elected course details---");
        System.out.println("Course name: "+courseName);
        System.out.println("Teacher name: "+teacherName);
        System.out.println("Classroom: "+classroom);
        System.out.println("---------------------------");
    }

    private static void insertStudent() throws IOException, ClassNotFoundException {
        System.out.print("input student name: ");
        name = scanner.next();
        System.out.print("input student sex(male or female): ");
        sex = scanner.next();
        System.out.print("input student age: ");
        age = scanner.nextInt();
        System.out.print("input student id: ");
        sid = scanner.next();
        System.out.print("input student major: ");
        major = scanner.next();
        Student student = new Student(name, sex, age, sid, major);
        file.writeFile(student);

        readFile(INSERT_STUDENT_CASE);
}

private static void insertTeacher() throws IOException, ClassNotFoundException {
        System.out.print("input teacher name: ");
        name = scanner.next();
        System.out.print("input teacher sex(male or female): ");
        sex = scanner.next();
        System.out.print("input teacher age: ");
        age = scanner.nextInt();
        System.out.print("input teacher id: ");
        tid = scanner.next();
        System.out.print("input teacher title: ");
        title = scanner.next();

        Teacher teacher = new Teacher(name, sex, age, tid, title);
        file.writeFile(teacher);

        readFile(INSERT_TEACHER_CASE);
}

private static void insertCourse() throws IOException, ClassNotFoundException {
        System.out.print("input course name: ");
        cname = scanner.next();
        System.out.print("input course id: ");
        cid = scanner.next();
        System.out.print("input course hour: ");
        chour = scanner.nextInt();

        Course course = new Course(cname, cid, chour);
        file.writeFile(course);

        readFile(INSERT_COURSE_CASE);
    }

    private static void insertSchedule() throws IOException, ClassNotFoundException {
        System.out.print("input class id: ");
        classid = scanner.next();
        System.out.print("input course id: ");
        cid = scanner.next();
        System.out.print("input teacher id: ");
        tid = scanner.next();
        System.out.print("input classroom: ");
        classroom = scanner.next();

        Schedule schedule = new Schedule(classid, cid, tid, classroom);
        file.writeFile(schedule);

        readFile(INSERT_SCHEDULE_CASE);
    }

    private static void insertElectivecourse() throws IOException, ClassNotFoundException {
        System.out.print("input class id: ");
        classid = scanner.next();
        System.out.print("input elective course id: ");
        elid = scanner.next();
        System.out.print("input student id: ");
        sid = scanner.next();

        Electivecourse electivecourse = new Electivecourse(classid, elid, sid);
        file.writeFile(electivecourse);

        readFile(INSERT_ELECTIVECOURSE_CASE);
    }
    // 启动系统时读取所有文件信息
    private static void readFile() throws IOException, ClassNotFoundException {
        file.readFile(students);
        file.readFile(teachers);
        file.readFile(courses);
        file.readFile(schedules);
        file.readFile(electivecourses);
    }

    // 添加信息后刷新并读取对应文件信息（增加性能）
    private static void readFile(int choice) throws IOException, ClassNotFoundException {
        switch (choice) {
            case INSERT_STUDENT_CASE:
                file.readFile(students);
                break;
            case INSERT_TEACHER_CASE:
                file.readFile(teachers);
                break;
            case INSERT_COURSE_CASE:
                file.readFile(courses);
                break;
            case INSERT_SCHEDULE_CASE:
                file.readFile(schedules);
                break;
            case INSERT_ELECTIVECOURSE_CASE:
                file.readFile(electivecourses);
                break;
            default:
                break;
        }
    }
}
```

实验类代码

代码 8-3 Person

```
import java.io.Serializable;

/**
 * @author Lolipop
 * @lastUpdate 2019/12/7
 */
public class Person implements Serializable {
    protected String name;
    protected String sex;
    int age;

    Person(String name, String sex, int age) {
        this.name = name;
        this.sex = sex;
        this.age = age;
    }

    String getName() {
        return this.name;
    }

    String getSex() {
        return this.sex;
    }

    String getAge() {
        return Integer.toString(this.age);
    }

    void setName(String name) {
        this.name = name;
    }
    void setSex(String sex) {
        this.sex = sex;
    }

    void setAge(int age) {
        this.age = age;
    }

    void display() {
        System.out.print("name: "+this.name+"\nsex: "+this.sex+"\nage: "+this.age+"\n");
    }
}
```

代码 8-4 Student

```
/**
 * @author Lolipop
 * @lastUpdate 2019/12/7
 */
public class Student extends Person{
    private String sid;
    private String major;

    Student(String name, String sex, int age, String sid, String major) {
        super(name, sex, age);
        this.sid = sid;
        this.major = major;
    }

    String getSid() {
        return this.sid;
    }

    String getMajor() {
        return this.major;
    }

    void setSid(String sid) {
        this.sid = sid;
    }

    void setMajor(String major) {
        this.major = major;
    }

    @Override
    void display() {
        System.out.print("name:    "+this.name+"\nsex:    "+this.sex+"\nage:    "+this.age+"\nsid:
```

```
"+this.sid+"\nmajor: "+this.major+"\n");
        }
}
```

代码 8-5 Teacher

```
/**
 * @author Lolipop
 * @lastUpdate 2019/12/7
 */
public class Teacher extends Person {
    private String tid;
    private String title;

    Teacher(String name, String sex, int age, String tid, String title) {
        super(name, sex, age);
        this.tid = tid;
        this.title = title;
    }

    String getTid() {
        return this.tid;
    }

    String getTitle() {
        return this.title;
    }

    void setTid(String tid) {
        this.tid = tid;
    }

    void setTitle(String title) {
        this.title = title;
    }

    @Override
    void display() {
        System.out.print("name: "+this.name+"\nsex: "+this.sex+"\nage: "+this.age+"\ntid:
```

```
"+this.tid+"\ntitle: "+this.title+"\n");
        }
}
```

代码 8-6 Course

```
import java.io.Serializable;

/**
 * @author Lolipop
 * @lastUpdate 2019/12/7
 */
public class Course implements Serializable {
    private String cname;
    private String cid;
    private int chour;

    Course(String cname, String cid, int chour) {
        this.cname = cname;
        this.cid = cid;
        this.chour = chour;
    }

    String getCname() {
        return this.cname;
    }

    String getCid() {
        return this.cid;
    }

    String getChour() {
        return Integer.toString(this.chour);
    }

    void setCname(String cname) {
        this.cname = cname;
    }
	void setCid(String cid) {
		this.cid = cid;
	}

	void setChour(int chour) {
		this.chour = chour;
	}

	void display() {
		System.out.print("cname: "+this.cname+"\ncid: "+this.cid+"\nchour: "+this.chour+"\n");
	}
}
```

代码 8-7 Schedule

```
import java.io.Serializable;

/**
 * @author Lolipop
 * @lastUpdate 2019/12/7
 */
public class Schedule implements Serializable {
    private String classid;
    private String cid;
    private String tid;
    private String classroom;

    Schedule(String classid, String cid, String tid, String classroom) {
        this.classid = classid;
        this.cid = cid;
        this.tid = tid;
        this.classroom = classroom;
    }

    String getClassid() {
        return this.classid;
    }

    String getCid() {
        return this.cid;
    }

    String getTid() {
        return this.tid;
    }

    String getClassroom() {
        return this.classroom;
	}

	void setClassid(String classid) {
		this.classid = classid;
	}

	void setTid(String tid) {
		this.tid = tid;
	}

	void setCid(String cid) {
		this.cid = cid;
	}

	void setClassroom(String classroom) {
		this.classroom = classroom;
	}

	void display() {
		System.out.print("classid:                    "+this.classid+"\ncid:                    "+this.cid+"\ntid:
"+this.tid+"\nclassroom: "+this.classroom+"\n");
	}
}
```

代码 8-8 Electivecourse

```
import java.io.Serializable;

/**
 * @author Lolipop
 * @lastUpdate 2019/12/7
 */
public class Electivecourse implements Serializable {
    private String elid;
    private String sid;
    private String classid;

    Electivecourse(String classid, String elid, String sid) {
        this.classid = classid;
        this.elid = elid;
        this.sid = sid;
    }

    String getClassid() {
        return this.classid;
    }

    String getElid() {
        return this.elid;
    }

    String getSid() {
        return this.sid;
    }

    void setClassid(String classid) {
        this.classid = classid;
    }
    void setElid(String elid) {
        this.elid = elid;
    }

    void setSid(String sid) {
        this.sid = sid;
    }

    void display() {
        System.out.print("elid: "+this.elid+"\nsid: "+this.sid+"\nclassid: "+this.classid+"\n");
    }
}
```

代码 8-9 Myfile

```
import java.io.*;

/**
 * @author Lolipop
 * @lastUpdate 2019/12/12
 */
public class Myfile {
	//  设置 opfile 默认路径
	File opfile;

	// file 存储的最多信息数，默认为 30 条
	int MAXSIZE = 30;

	Myfile(String filepath) {
		opfile = new File(filepath);
	}

	File getFile() {
		//返回存储数据文件的目录
		return opfile;
	}

	int setMAXSIZE(int size) {
		return this.MAXSIZE = size;
	}

	// StudentCase
	void writeFile(Student student) throws IOException, ClassNotFoundException {
		File file = new File(opfile, "studentData.dat");
		Student[] writeStudents;

		if (file.exists() && file.length()!=0) {
			//  若文件存在，则先从文件中读取对象到对象数组 readStudents 中
```

```
            FileInputStream fis = new FileInputStream(file);
            ObjectInputStream ois = new ObjectInputStream(fis);
            Student[] readStudents = (Student[]) ois.readObject();
            ois.close();
            fis.close();

            // 将 readStudents 复制给 writeStudents
            // 并在 writeStudents 的末尾加上新加对象 student
            writeStudents = new Student[readStudents.length+1];
            System.arraycopy(readStudents, 0, writeStudents, 0, readStudents.length);
            writeStudents[readStudents.length] = student;
        } else {
            // 若文件不存在，则把 writeStudents 的第一项赋为对象 student
            writeStudents = new Student[1];
            writeStudents[0] = student;
        }

        // 将对象数组 writeStudents 序列化并覆盖原文件
        FileOutputStream fos = new FileOutputStream(file);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(writeStudents);
        oos.flush();
        oos.close();
        fos.close();
    }

    void readFile(Student[] students) throws IOException, ClassNotFoundException {
        File file = new File(opfile, "studentData.dat");

        // 文件不存在时直接返回 null
        if (!file.exists()) {
            return;
        }
        //  文件存在时，读取文件信息到对象数组中
        FileInputStream fis = new FileInputStream(file);
        ObjectInputStream ois = new ObjectInputStream(fis);
        Student[] readStudents = (Student[]) ois.readObject();
        System.arraycopy(readStudents, 0, students, 0, readStudents.length);
        ois.close();
        fis.close();
}

// TeacherCase
void writeFile(Teacher teacher) throws IOException, ClassNotFoundException {
        File file = new File(opfile, "teacherData.dat");
        Teacher[] writeTeachers;

        if (file.exists() && file.length()!=0) {
                FileInputStream fis = new FileInputStream(file);
                ObjectInputStream ois = new ObjectInputStream(fis);
                Teacher[] readTeachers = (Teacher[]) ois.readObject();
                ois.close();
                fis.close();

                writeTeachers = new Teacher[readTeachers.length+1];
                System.arraycopy(readTeachers, 0, writeTeachers, 0, readTeachers.length);
                writeTeachers[readTeachers.length] = teacher;
        } else {
                writeTeachers = new Teacher[1];
                writeTeachers[0] = teacher;
        }

        FileOutputStream fos = new FileOutputStream(file);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(writeTeachers);
        oos.flush();
        oos.close();
        fos.close();
    }

    void readFile(Teacher[] teachers) throws IOException, ClassNotFoundException {
        File file = new File(opfile, "teacherData.dat");

        if (!file.exists()) {
            return;
        }

        FileInputStream fis = new FileInputStream(file);
        ObjectInputStream ois = new ObjectInputStream(fis);
        Teacher[] readTeachers = (Teacher[]) ois.readObject();
        System.arraycopy(readTeachers, 0, teachers, 0, readTeachers.length);
        ois.close();
        fis.close();
    }

    // CourseCase
    void writeFile(Course course) throws IOException, ClassNotFoundException {
        File file = new File(opfile, "courseData.dat");
        Course[] writeCourses;

        if (file.exists() && file.length()!=0) {
            FileInputStream fis = new FileInputStream(file);
            ObjectInputStream ois = new ObjectInputStream(fis);
            Course[] readCourses = (Course[]) ois.readObject();
            ois.close();
            fis.close();

            writeCourses = new Course[readCourses.length+1];
            System.arraycopy(readCourses, 0, writeCourses, 0, readCourses.length);
            writeCourses[readCourses.length] = course;
        } else {
            writeCourses = new Course[1];
            writeCourses[0] = course;
        }

        FileOutputStream fos = new FileOutputStream(file);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(writeCourses);
        oos.flush();
        oos.close();
        fos.close();
    }

    void readFile(Course[] courses) throws IOException, ClassNotFoundException {
        File file = new File(opfile, "courseData.dat");

        if (!file.exists()) {
            return;
        }

        FileInputStream fis = new FileInputStream(file);
        ObjectInputStream ois = new ObjectInputStream(fis);
        Course[] readCourses = (Course[]) ois.readObject();
        System.arraycopy(readCourses, 0, courses, 0, readCourses.length);
        ois.close();
        fis.close();
    }

    // ScheduleCase
    void writeFile(Schedule schedule) throws IOException, ClassNotFoundException {
        File file = new File(opfile, "scheduleData.dat");
        Schedule[] writeSchedules;

        if (file.exists() && file.length()!=0) {
            FileInputStream fis = new FileInputStream(file);
            ObjectInputStream ois = new ObjectInputStream(fis);
            Schedule[] readSchedules = (Schedule[]) ois.readObject();
            ois.close();
            fis.close();

            writeSchedules = new Schedule[readSchedules.length+1];
            System.arraycopy(readSchedules, 0, writeSchedules, 0, readSchedules.length);
            writeSchedules[readSchedules.length] = schedule;
        } else {
            writeSchedules = new Schedule[1];
            writeSchedules[0] = schedule;
        }

        FileOutputStream fos = new FileOutputStream(file);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(writeSchedules);
        oos.flush();
        oos.close();
        fos.close();
    }

    void readFile(Schedule[] schedules) throws IOException, ClassNotFoundException {
        File file = new File(opfile, "scheduleData.dat");

        if (!file.exists()) {
            return;
        }

        FileInputStream fis = new FileInputStream(file);
        ObjectInputStream ois = new ObjectInputStream(fis);
        Schedule[] readSchedules = (Schedule[]) ois.readObject();
        System.arraycopy(readSchedules, 0, schedules, 0, readSchedules.length);
        ois.close();
        fis.close();
```

```
	}

	// ElectivecourseCase
	void writeFile(Electivecourse electivecourse) throws IOException, ClassNotFoundException
{
		File file = new File(opfile, "electivecourseData.dat");
		Electivecourse[] writeElectivecourses;

		if (file.exists() && file.length()!=0) {
			FileInputStream fis = new FileInputStream(file);
			ObjectInputStream ois = new ObjectInputStream(fis);
			Electivecourse[] readElectivecourses = (Electivecourse[]) ois.readObject();
			ois.close();
			fis.close();

			writeElectivecourses = new Electivecourse[readElectivecourses.length+1];
			System.arraycopy(readElectivecourses, 0, writeElectivecourses, 0,
readElectivecourses.length);
			writeElectivecourses[readElectivecourses.length] = electivecourse;
		} else {
			writeElectivecourses = new Electivecourse[1];
			writeElectivecourses[0] = electivecourse;
		}

		FileOutputStream fos = new FileOutputStream(file);
		ObjectOutputStream oos = new ObjectOutputStream(fos);
		oos.writeObject(writeElectivecourses);
		oos.flush();
		oos.close();
		fos.close();
	}

	void readFile(Electivecourse[] electivecourses) throws IOException,
ClassNotFoundException {
        File file = new File(opfile, "electivecourseData.dat");

        if (!file.exists()) {
            return;
        }

        FileInputStream fis = new FileInputStream(file);
        ObjectInputStream ois = new ObjectInputStream(fis);
        Electivecourse[] readElectivecourses = (Electivecourse[]) ois.readObject();
        System.arraycopy(readElectivecourses,        0,        electivecourses,        0,
readElectivecourses.length);
        ois.close();
        fis.close();
    }
}
```

测试用代码

代码 8-10 TestPerson

```
/**
 * @author Lolipop
 * @lastUpdate 2019/12/7
 */
public class TestPerson {
    public static void main(String[] args) {
        Person testPerson = new Person("XiaoMing", "Male", 20);
        testPerson.display();
        System.out.println("-----------------");

        testPerson.setName("XiaoHong");
        testPerson.display();
        System.out.println("-----------------");

        testPerson.setSex("Female");
        testPerson.display();
        System.out.println("-----------------");

        testPerson.setAge(18);
        testPerson.display();
        System.out.println("-----------------");

        String info = "name: "+testPerson.getName()+"\nsex: "+testPerson.getSex()+"\nage: "+testPerson.getAge();
        System.out.print(info);
    }
}
```

代码 8-11 TestStudent

```
/**
  * @author Lolipop
  * @lastUpdate 2019/12/9
  */
public class TestStudent {
        public static void main(String[] args) {
                Student testStudent = new Student("XiaoMing", "Male", 20, "2018091202000",
"Computer");
                testStudent.display();
                System.out.println("-----------------");

                testStudent.setSid("2019091203024");
                testStudent.display();
                System.out.println("-----------------");

                testStudent.setMajor("Design");
                testStudent.display();
                System.out.println("-----------------");

                String info = "sid: "+testStudent.getSid()+"\nmajor: "+testStudent.getMajor();
                System.out.print(info);
        }
}
```

代码 8-12 TestTeacher

```
/**
 * @author Lolipop
 * @lastUpdate 2019/12/7
 */
public class TestTeacher {
    public static void main(String[] args) {
        Teacher testTeacher = new Teacher("XiaoMing", "Male", 20, "10001", "professor");
        testTeacher.display();
        System.out.println("-----------------");

        testTeacher.setTid("10005");
        testTeacher.display();
        System.out.println("-----------------");

        testTeacher.setTitle("associate professor");
        testTeacher.display();
        System.out.println("-----------------");

        String info = "tid: "+testTeacher.getTid()+"\ntitle: "+testTeacher.getTitle();
        System.out.print(info);
    }
}
```

代码 8-13 TestCourse

```
/**
 * @author Lolipop
 * @lastUpdate 2019/12/7
 */
public class TestCourse {
    public static void main(String[] args) {
        Course testCourse = new Course("Java", "2019001", 48);
        testCourse.display();
        System.out.println("-----------------");

        testCourse.setChour(96);
        testCourse.setCid("2019002");
        testCourse.setCname("Cpp");
        testCourse.display();
        System.out.println("-----------------");

        String info = "cname: "+testCourse.getCname()+"\ncid: "+testCourse.getCid()+"\nchour: "+testCourse.getChour();
        System.out.print(info);
    }
}
```

代码 8-14 TestSchedule

```
/**
 * @author Lolipop
 * @lastUpdate 2019/12/7
 */
public class TestSchedule {
    public static void main(String[] args) {
        Schedule testSchedule = new Schedule("2018091202", "2018002", "10004", "xy02");
        testSchedule.display();
        System.out.println("-----------------");

        testSchedule.setClassid("2019091202");
        testSchedule.setClassroom("xy04");
        testSchedule.setCid("2018004");
        testSchedule.setTid("10005");
        testSchedule.display();
        System.out.println("-----------------");

        String info = "classid: "+testSchedule.getClassid()+"\ncid: "+testSchedule.getCid()+"\ntid: "+testSchedule.getTid()+"\nclassroom: "+testSchedule.getClassroom();
        System.out.print(info);
    }
}
```

代码 8-15 TestElcourse

```
/**
 * @author Lolipop
 * @lastUpdate 2019/12/7
 */
public class TestElcourse {
    public static void main(String[] args) {
        Electivecourse testElcourse = new Electivecourse("2018091202", "1",
"2018091202000");
        testElcourse.display();
        System.out.println("-----------------");

        testElcourse.setClassid("2018091609");
        testElcourse.setElid("2");
        testElcourse.setSid("2018091609000");
        testElcourse.display();
        System.out.println("-----------------");

        String info = "elid: "+testElcourse.getElid()+"\nsid: "+testElcourse.getSid()+"\nclassid:
"+testElcourse.getClassid();
        System.out.print(info);
    }
}
```

代码 8-16 TestMyfile

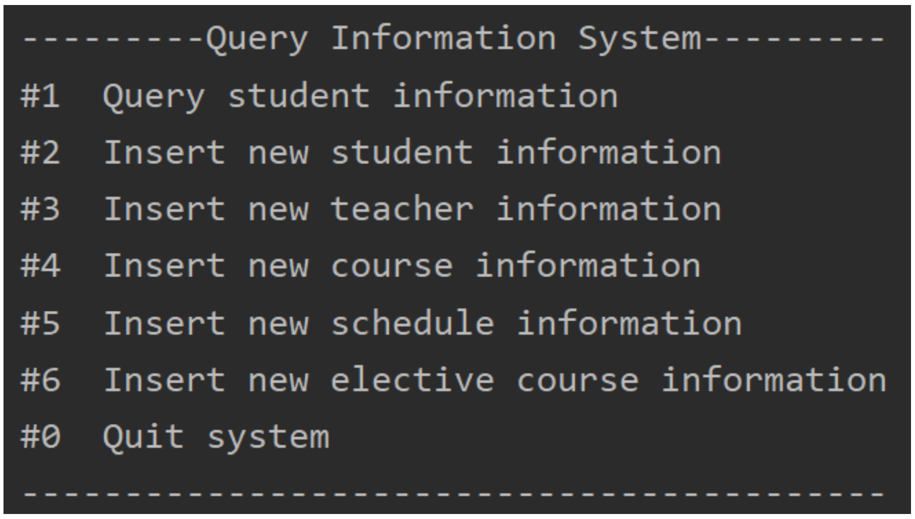
```
import java.io.*;


/**
 * @author Lolipop
 * @lastUpdate 2019/12/8
 */
public class TestMyfile {
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        // 在这里修改测试文件保存目录
        Myfile file = new Myfile("H:\\Lolipop");

        // 测试例
        Student[] students = new Student[2];
        students[0] = new Student("XiaoMing", "male", 18, "2018091202000", "computer");
        students[1] = new Student("XiaoHong", "female", 20, "2018091202001", "design");
        file.writeFile(students[0]);
        file.writeFile(students[1]);
        file.writeFile(students[0]);
    }
}
```

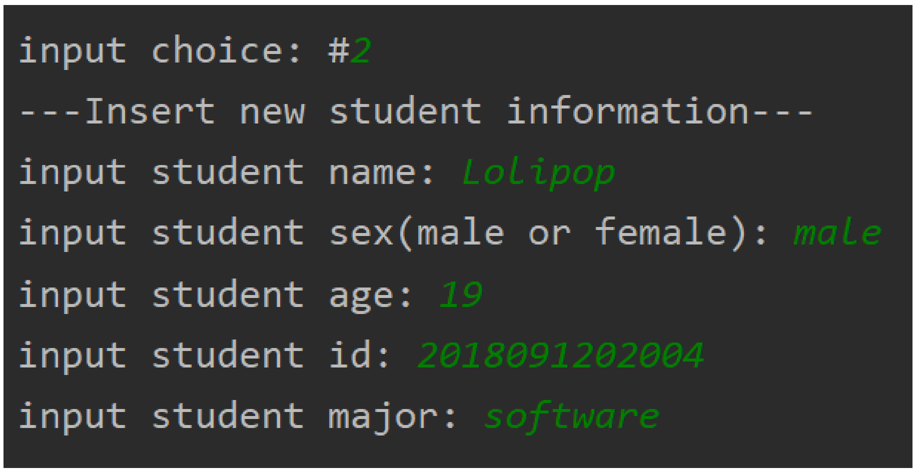
九、程序运行结果：

实验运行结果



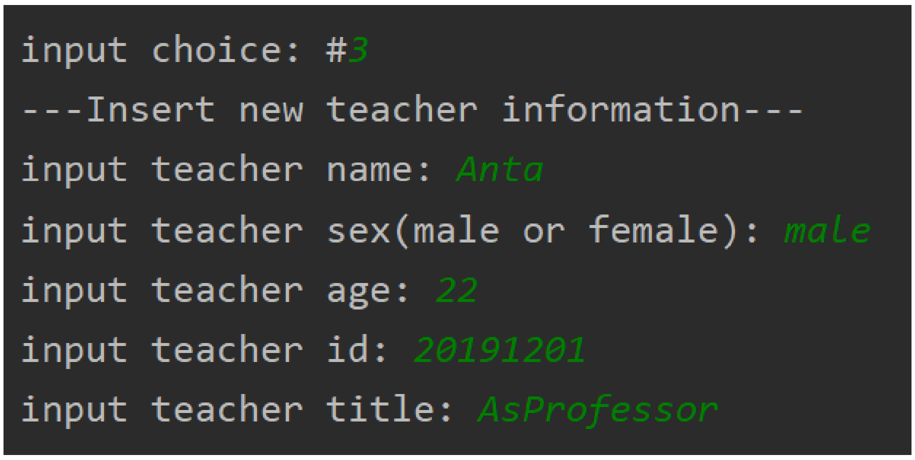
```
---------Query Information System---------
#1  Query student information
#2  Insert new student information
#3  Insert new teacher information
#4  Insert new course information
#5  Insert new schedule information
#6  Insert new elective course information
#0  Quit system
------------------------------------------
```

图 9-1 主界面



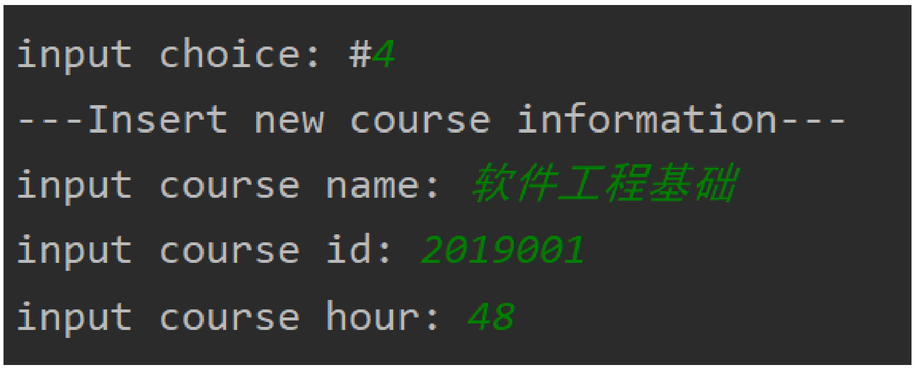
```
input choice: #2
---Insert new student information---
input student name: Lolipop
input student sex(male or female): male
input student age: 19
input student id: 2018091202004
input student major: software
```

图 9-2 添加学生信息



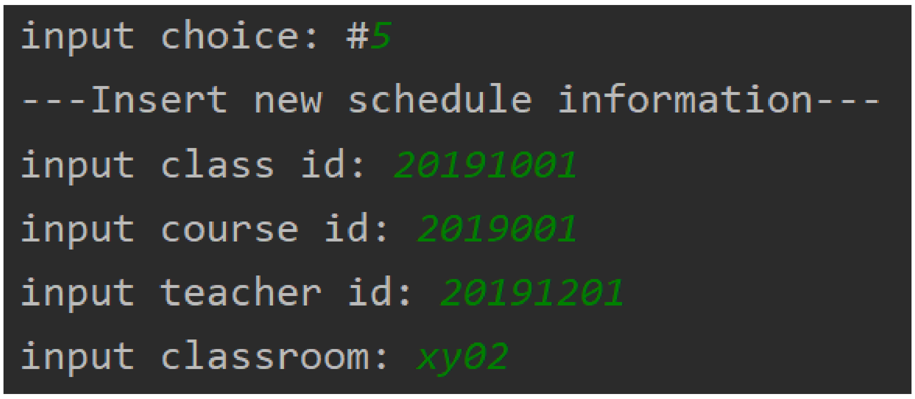
```
input choice: #3
---Insert new teacher information---
input teacher name: Anta
input teacher sex(male or female): male
input teacher age: 22
input teacher id: 20191201
input teacher title: AsProfessor
```

图 9-3 添加教师信息



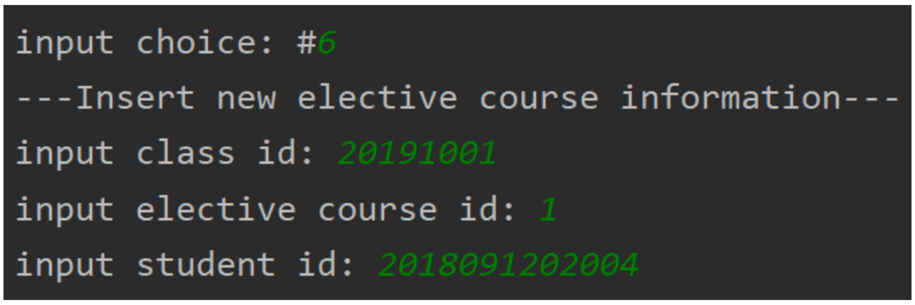
```
input choice: #4
---Insert new course information---
input course name: 软件工程基础
input course id: 2019001
input course hour: 48
```

图 9-4 添加课程信息



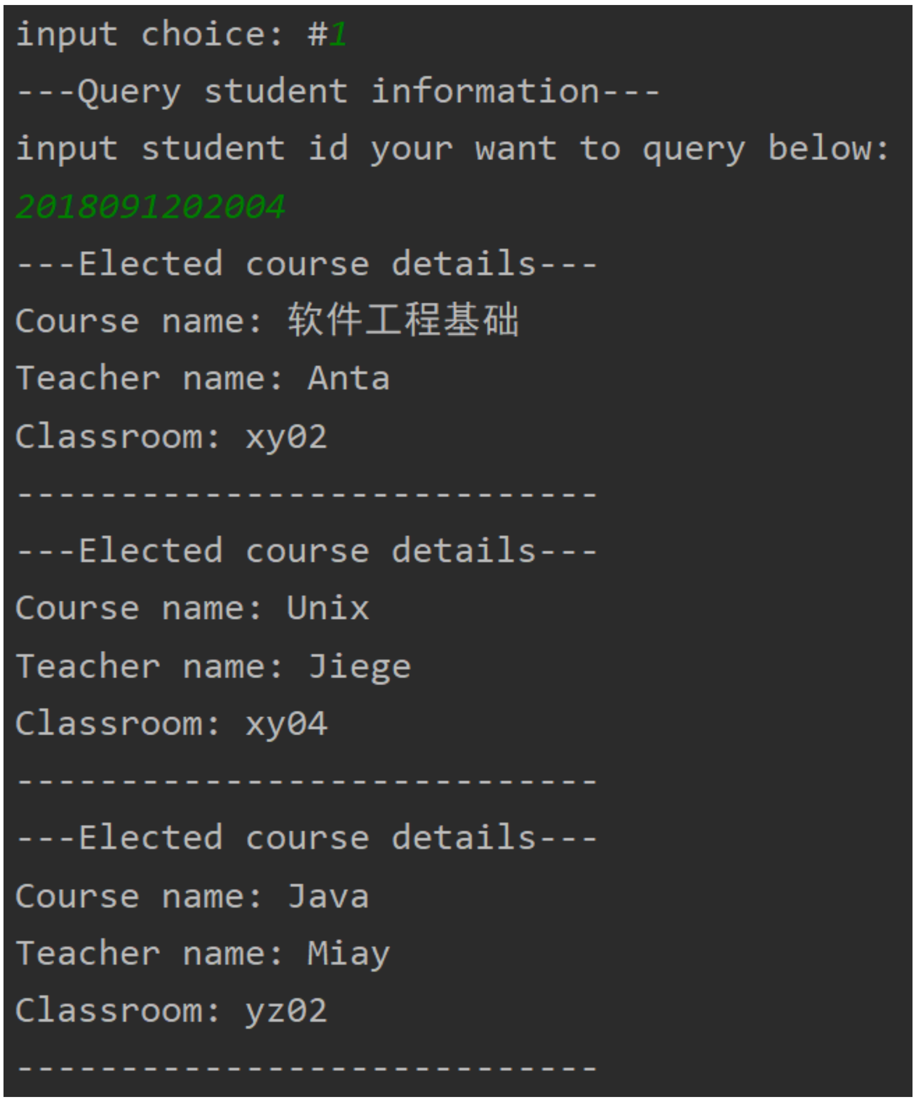
```
input choice: #5
---Insert new schedule information---
input class id: 20191001
input course id: 2019001
input teacher id: 20191201
input classroom: xy02
```

图 9-5 添加排课信息



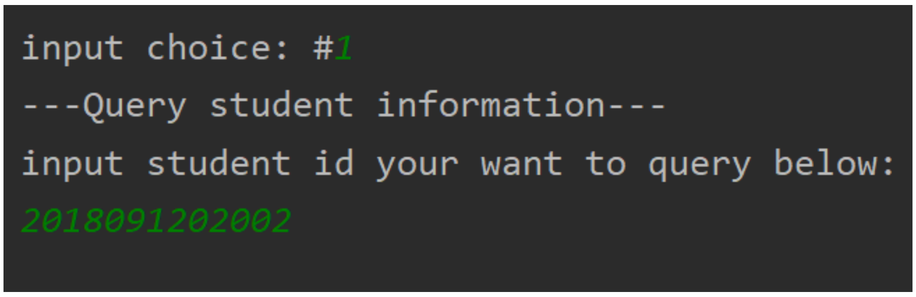
```
input choice: #6
---Insert new elective course information---
input class id: 20191001
input elective course id: 1
input student id: 2018091202004
```

图 9-6 添加选课信息



```
input choice: #1
---Query student information---
input student id your want to query below:
2018091202004
---Elected course details---
Course name: 软件工程基础
Teacher name: Anta
Classroom: xy02
----------------------------
---Elected course details---
Course name: Unix
Teacher name: Jiege
Classroom: xy04
----------------------------
---Elected course details---
Course name: Java
Teacher name: Miay
Classroom: yz02
----------------------------
```

图 9-7 查询选课信息（三个信息）

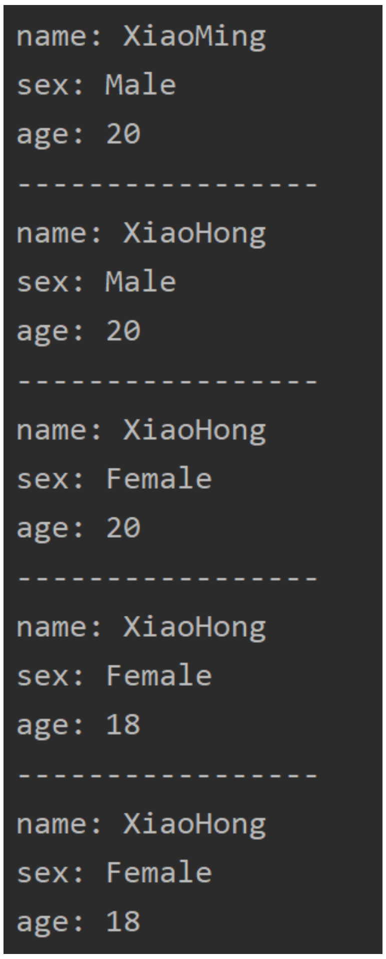


```
input choice: #1
---Query student information---
input student id your want to query below:
2018091202002
```

图 9-8 查询选课信息（无结果）

测试运行结果

1. 新建一个 Person 对象并打印，接下来分别修改对象的 name、sex、age 变量并打印，最后测试对象的 getName()、getSex()、getAge()方法。结果如图 9-9 所示。

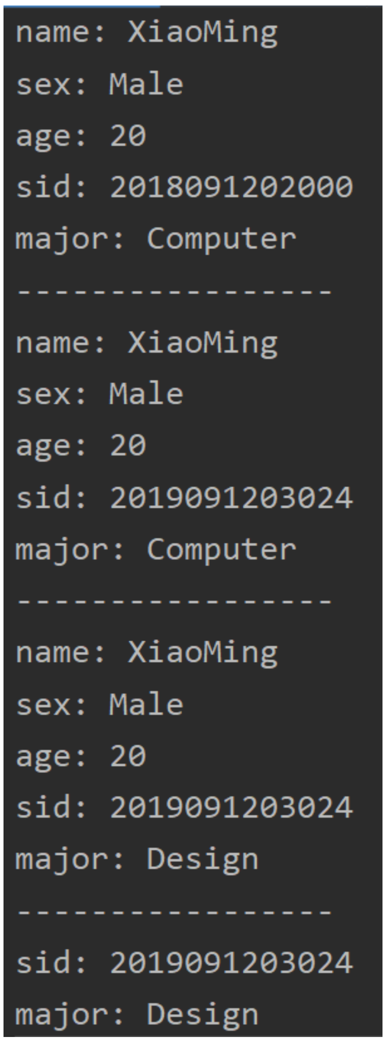


```
name: XiaoMing
sex: Male
age: 20
-----------------
name: XiaoHong
sex: Male
age: 20
-----------------
name: XiaoHong
sex: Female
age: 20
-----------------
name: XiaoHong
sex: Female
age: 18
-----------------
name: XiaoHong
sex: Female
age: 18
```

图 9-9 TestPerson 运行结果

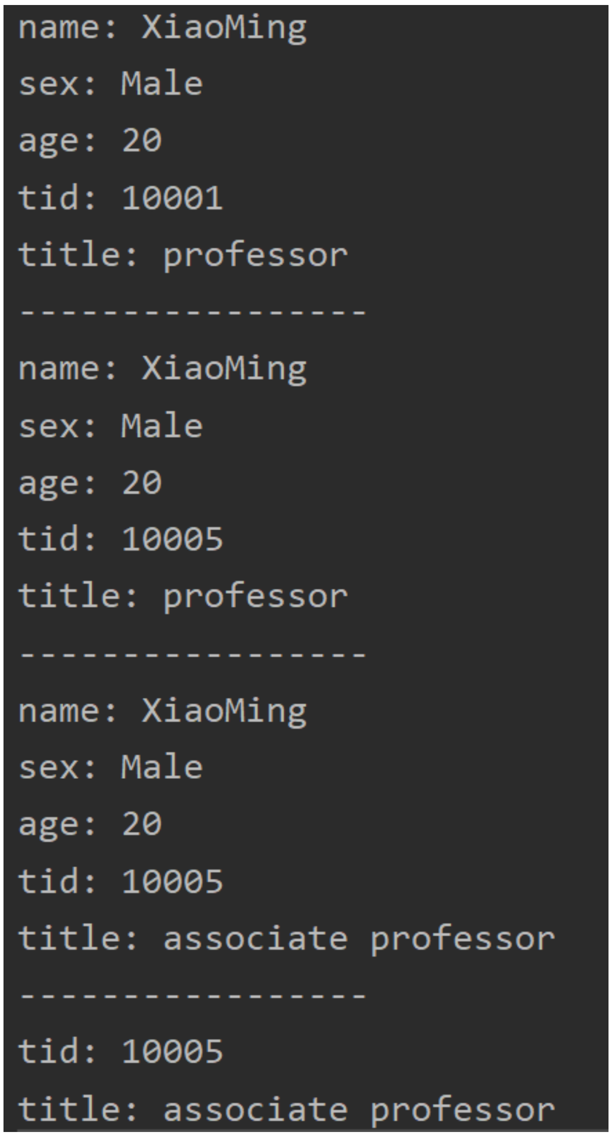
2. 新建一个 Student 对象并打印，接下来分别修改对象的 sid、major 变量并打印，最后测试对象的 getSid()、getMajor()方法。结果如图 9-10 所示。
3. 新建一个 Teacher 对象并打印，接下来分别修改对象的 tid、title 变量并打印，最后测试对象的 getTid()、getTitle()方法。结果如图 9-11 所示。
4. 新建一个 Course 对象并打印，接下来修改对象的 chour、cid、cname 变量并打印，最后测试对象的 getCname ()、getCid ()、getChour()方法。结果如图 9-12 所示。

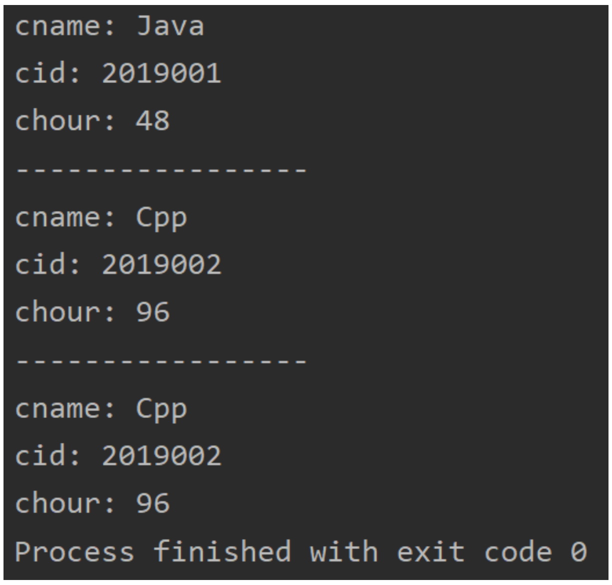
5. 新建一个 Schedule 对象并打印，接下来修改对象的 classid、classroom、cid、tid 变量并打印，最后测试对象的 getClassid ()、getCid ()、getTid ()、getClassroom()方法。结果如图 9-13 所示。
6. 新建一个 Electivecourse 对象并打印，接下来修改对象的 classid、elid、sid 变量并打印，最后测试对象的 getElid ()、getSid ()、getClassid ()方法。结果如图 9-14 所示。
7. 新建一个 Myfile 对象，新建对象数组 Student[]并赋值，调用 writeFile(Student student)方法将对象序列化存储到文件中。保存后的文件如图 9-15 所示。

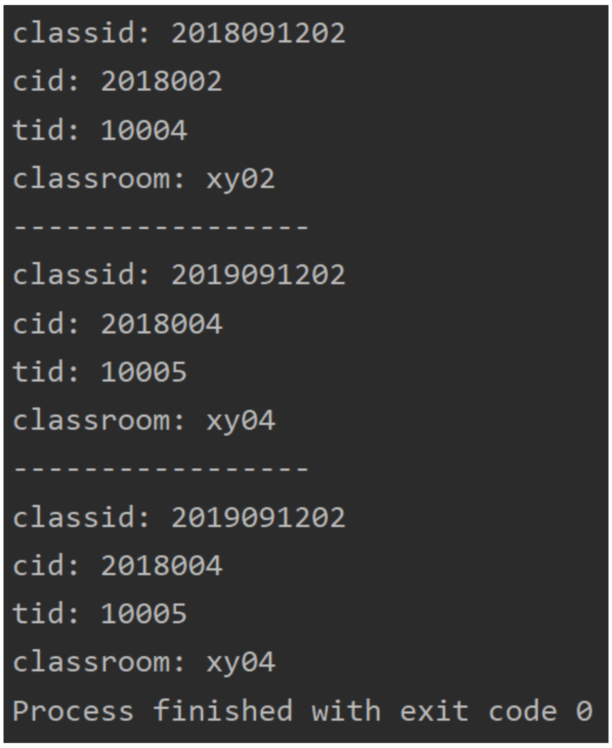


```
name: XiaoMing
sex: Male
age: 20
sid: 2018091202000
major: Computer
-----------------
name: XiaoMing
sex: Male
age: 20
sid: 2019091203024
major: Computer
-----------------
name: XiaoMing
sex: Male
age: 20
sid: 2019091203024
major: Design
-----------------
sid: 2019091203024
major: Design
```

图 9-10 TestStudent 运行结果

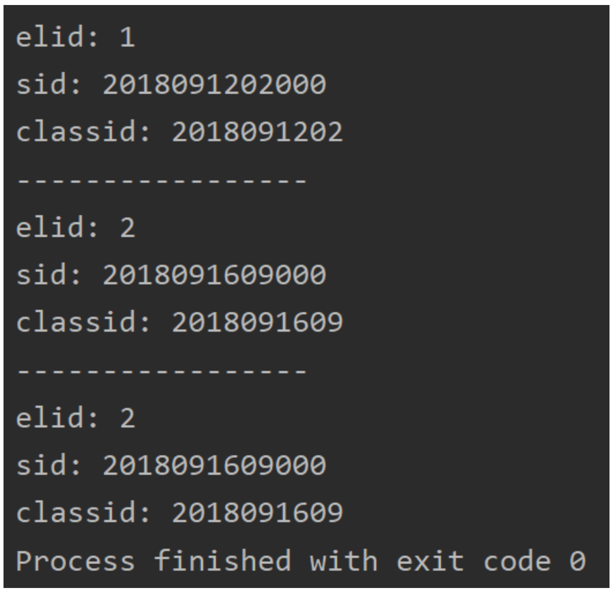

```
name: XiaoMing
sex: Male
age: 20
tid: 10001
title: professor
-----------------
name: XiaoMing
sex: Male
age: 20
tid: 10005
title: professor
-----------------
name: XiaoMing
sex: Male
age: 20
tid: 10005
title: associate professor
-----------------
tid: 10005
title: associate professor
```
图 9-11 TestTeacher 运行结果


```
cname: Java
cid: 2019001
chour: 48
-----------------
cname: Cpp
cid: 2019002
chour: 96
-----------------
cname: Cpp
cid: 2019002
chour: 96
Process finished with exit code 0
```
图 9-12 TestCourse 运行结果



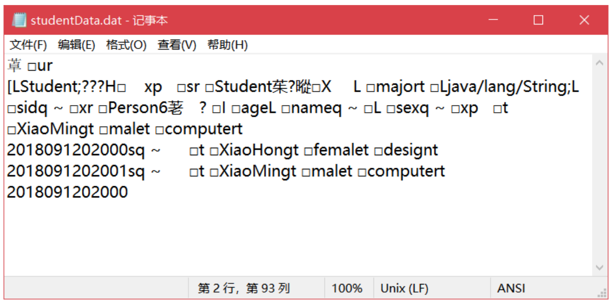
```
classid: 2018091202
cid: 2018002
tid: 10004
classroom: xy02
-----------------
classid: 2019091202
cid: 2018004
tid: 10005
classroom: xy04
-----------------
classid: 2019091202
cid: 2018004
tid: 10005
classroom: xy04
Process finished with exit code 0
```

图 9-13 TestSchedule 运行结果



```
elid: 1
sid: 2018091202000
classid: 2018091202
-----------------
elid: 2
sid: 2018091609000
classid: 2018091609
-----------------
elid: 2
sid: 2018091609000
classid: 2018091609
Process finished with exit code 0
```

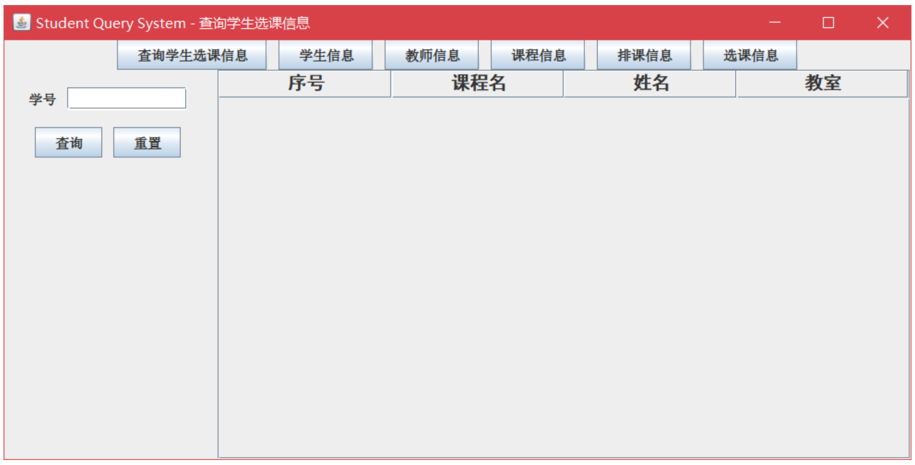
图 9-14 TestElcourse 运行结果



```
草 □ur
[LStudent;???H□   xp   □sr □Student桀?瞛□X     L □majort □Ljava/lang/String;L
□sidq ~ □xr □Person6蒈   ? □I □ageL □nameq ~ □L □sexq ~ □xp   □t
□XiaoMingt □malet □computert
2018091202000sq ~      □t □XiaoHongt □femalet □designt
2018091202001sq ~      □t □XiaoMingt □malet □computert
2018091202000
```

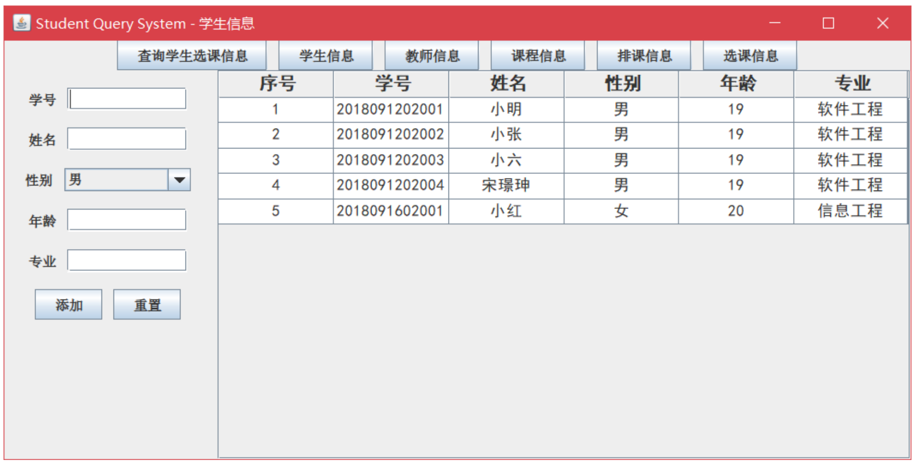
图 9-15 TestMyfile 运行结果

图形界面运行结果



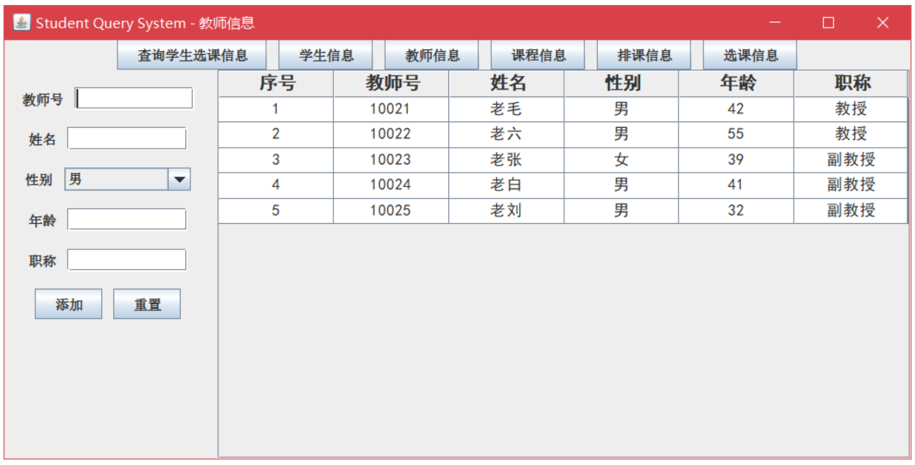
序号	课程名	姓名	教室

图 9-16 GUI 界面



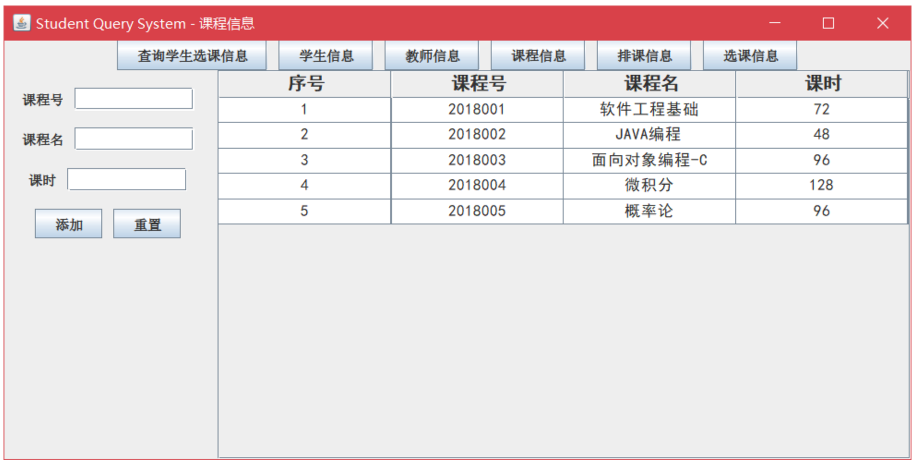
序号	学号	姓名	性别	年龄	专业
1	2018091202001	小明	男	19	软件工程
2	2018091202002	小张	男	19	软件工程
3	2018091202003	小六	男	19	软件工程
4	2018091202004	宋璟珅	男	19	软件工程
5	2018091602001	小红	女	20	信息工程

图 9-17 GUI-学生信息界面



序号	教师号	姓名	性别	年龄	职称
1	10021	老毛	男	42	教授
2	10022	老六	男	55	教授
3	10023	老张	女	39	副教授
4	10024	老白	男	41	副教授
5	10025	老刘	男	32	副教授

图 9-18 GUI-教师信息界面



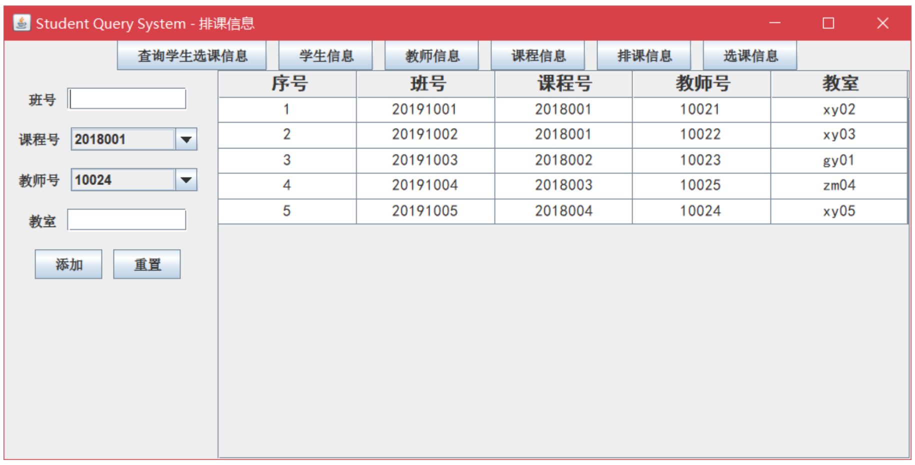
课程号

课程名

课时

序号	课程号	课程名	课时
1	2018001	软件工程基础	72
2	2018002	JAVA编程	48
3	2018003	面向对象编程-C	96
4	2018004	微积分	128
5	2018005	概率论	96

图 9-19 GUI-课程信息界面



班号

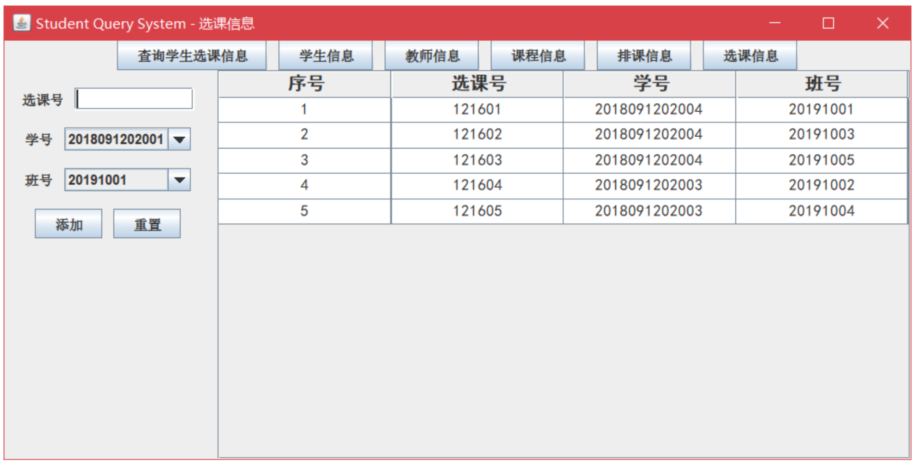
课程号 2018001

教师号 10024

教室

序号	班号	课程号	教师号	教室
1	20191001	2018001	10021	xy02
2	20191002	2018001	10022	xy03
3	20191003	2018002	10023	gy01
4	20191004	2018003	10025	zm04
5	20191005	2018004	10024	xy05

图 9-20 GUI-排课信息界面



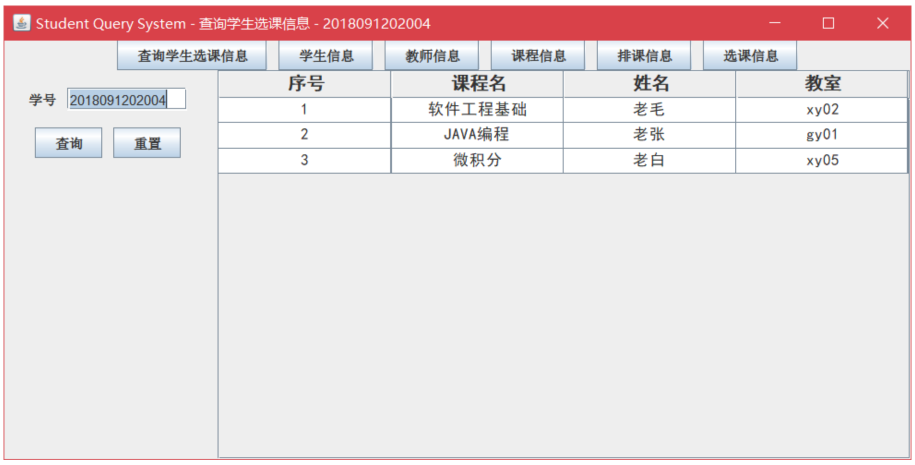
选课号

学号 2018091202001

班号 20191001

序号	选课号	学号	班号
1	121601	2018091202004	20191001
2	121602	2018091202004	20191003
3	121603	2018091202004	20191005
4	121604	2018091202003	20191002
5	121605	2018091202003	20191004

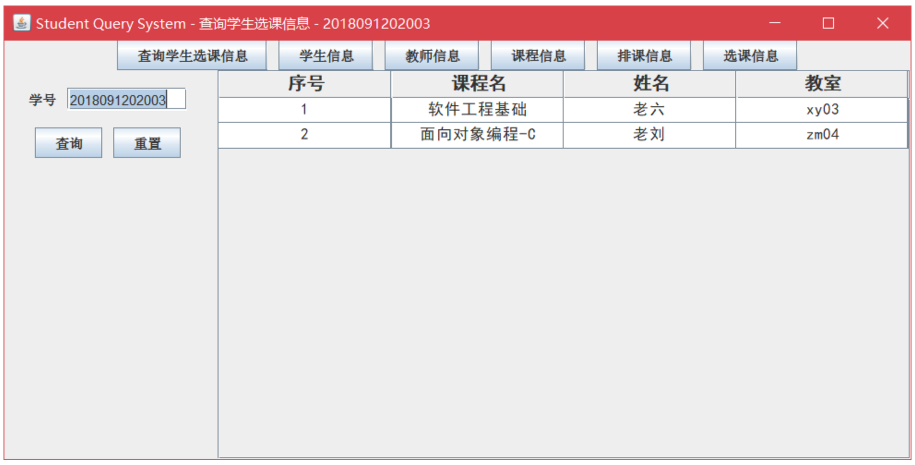
图 9-21 GUI-选课信息界面


Student Query System - 查询学生选课信息 - 2018091202004

2018091202004

序号	课程名	姓名	教室
1	软件工程基础	老毛	xy02
2	JAVA编程	老张	gy01
3	微积分	老白	xy05

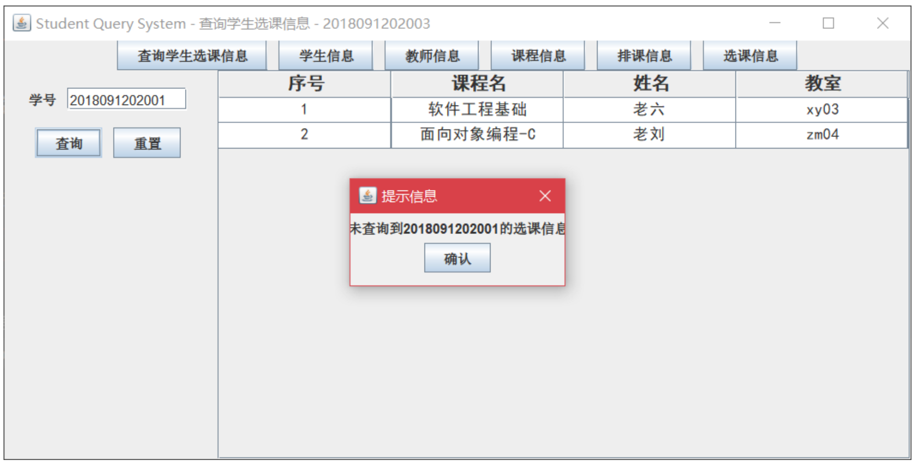
图 9-22 GUI-查询到 3 个选课信息


Student Query System - 查询学生选课信息 - 2018091202003

2018091202003

序号	课程名	姓名	教室
1	软件工程基础	老六	xy03
2	面向对象编程-C	老刘	zm04

图 9-23 GUI-查询到 2 个选课信息


Student Query System - 查询学生选课信息 - 2018091202003

2018091202001

序号	课程名	姓名	教室
1	软件工程基础	老六	xy03
2	面向对象编程-C	老刘	zm04

未查询到2018091202001的选课信息

图 9-24 GUI-未查询到选课信息