**天津工业大学**

**计算机科学与软件学院**

**实习报告**

课 程 名 称: 面向对象应用开发实习

实 习 题 目:基于MVC架构的GUI界面信息管理统

专 业 班 级: 计算机16024

学 生 姓 名: 汤应薇

学　　 号: 1610110044

实 习 成 绩:

指导教师签名： 年 月 日

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 实习时间 | | 2017.12.25-2018.  01.03 | 实习地点 | 软件园C205  C210 | 实习学时 | 30 |
| 实习  目的 | 1. 掌握java集合对象的基本用法 2. 掌握java泛型的基本处理机制 3. 掌握IO流的基本数据处理 4. 掌握 ArrayList等 Java 集合对象的基本用法 5. 掌握掌握 Java 中的基本文件 IO 与对象序列化方法 6. 掌握GUI图形界面接口 7. 掌握MVC架构   让学生通过这些针对性练习充分理解面向对象程序设计的思想及其应用方式。 | | | | | |
| 实习  内容 | 1. 第一阶段数组实现人员信息管理系统  2.利用 java 中的集合类 ArrayList 实现人员信息管理系统  3.对程序进行重构，将不同功能的类组织到不同的包中，增加模糊查询  4.利用IO对数据进行存取  5.利用MVC+GUI实现学生信息管理 | | | | | |
| 实  习  要  求 | 第一阶段（V1）利用数组实现管理系统；V2.1利用ArrayList实现第一阶段的内容，并进行重构，引入表示增删改查以及模糊查询等操作的Operate 接口；V2.2利用IO实现集合的存储和提取并抛出自定义异常V3利用文本文件实现数据的持久化；V4利用GUI写出图形界面，用MVC架构管理系统。 | | | | | |
| 实习  主要  设施  设备 | 开发环境：JDK 1.8.0\_152  编程软件：eclipse | | | | | |
| 实  习  成  果 | 1. 利用数组了实现人员信息管理系统 2. 利用集合类 ArrayList 实现人员信息管理系统，并进行了重构，不仅有增删改查功能还有模糊查询的功能，可以抛出自定义异常，利用IO实现了数据的存取并用文本文件实现了数据的持久化。 3. 用GUI写出图形界面，用MVC架构管理系统。 | | | | | |

|  |
| --- |
| **1 引言**  基于MVC架构的GUI界面信息管理统  1.1系统概述  实现一个简单的人员管理系统，完成人员信息的增删改查（两种查询），将对象保存在 Java 集合对象中，保存到本地的文本文档。  利用GUI写出图形界面，用MVC架构管理系统。  1.2 需求分析  通过本系统软件，能帮助教职工利用计算机，快速方便的对学生以及职工信息进行管理、输入、输出、查找、删除和修改与更新，使散乱的的人事档案能够具体、直观。  1.3 运行环境  开发环境：JDK 1.8.0\_152  编程软件：eclipse  **2 概要设计**  2.1 设计思路  首先实现学生信息管理系统功能（增删改查等）。  先利用MVC定义updateel层（添加各类方法），再使用定义controller调用updateel层的方法。  设计两个GUI主界面，添加所需按钮和文本区，并添加要实现的功能选项，然后对这些选项进行监听，实现其功能。  2.2 模块功能介绍（针对MVC三层分别介绍）  View层：Gui.java实现生成与用户交互的图形界面，调用controller中的方法实现相应的功能  Controller层：调用updateel层中的方法，函数，传递到Gui里中，供管理者使用。  updateel层：调用相应的函数给controller层，处理用户的请求。  2.3 模块结构图  View层：      创建图形界面  实现增删改查等功能  Control层：~O_[6@LEPEMN~B}OV`5[6AY  Model层：    2.4 程序流程图  接受用户数据并实现监听  调用controller类，处理并对相应的功能进行实现  调用Model类，调用相关的方法  返回数据  显示在图形界面相应的位置  **3　详细设计**（小三黑体，居中）  InfoSys应用包括三个核心接口。  View：视图层的接口，负责生成与用户交互的图形界面。  updateel: 控制器层的接口，负责调用模型和视图。  Controller: 模型层的接口，负责处理业务逻辑，访问文件。  如例程1 所示是 View接口的源程序。它包括以下 3 个方法。  ◆add UserGestureListener(Controller ctrl)方法：在视图中注册处理各种用户动作（比如用户按下【查询客户】按钮）的控制器，参数 ctrl 指定控制器。  ◆showDisplay(Object display)方法：在图形界面上显示数据，参数 display 指定待显示  的数据。  ◆handleStudentChange ()方法：当模型层修改了文件中某个客户的信息时，同步刷新  视图的图形界面。  例程1：  package view;  import java.rmi.Remote;  import java.rmi.RemoteException;  import InfoSys.InfoSysException;  import InfoSys.controller.InfoSysController\_stu;  import InfoSys.vo.Student;  public interface View extends Remote  {  /\*\*注册处理用户动作的监听器\*/  public void addUserGestureListener(Controller ctrl)throws InfoSysException,RemoteException;  /\*\*在图形界面上显示数据，参数display指定显示的数据\*/  public void showDisplay(Object display)throws InfoSysException,RemoteException ;  public void handleStudentChange(Student cust)throws InfoSysException,RemoteException;  }  例程2：Controller.java  package InfoSys.controller;  import java.util.List;  import InfoSys.vo.Student;  public interface Controller {  /\*\* 处理添加学生的动作 \*/  public void handleAddGesture(Student stu);  /\*\* 处理刷新学生的动作 \*/  public void handleShowGesture(Student stu) throws Exception;  /\*\* 处理根据ID查询学生的动作 \*/  public void handleFindByIDGesture(String id) throws Exception;  /\*\* 处理根据名字查询学生的动作 \*/  public void handleFindByNameGesture(String name) throws Exception;  /\*\* 处理删除学生的动作 \*/  public void handleDeleteGesture(Student stu) throws Exception;  /\*\* 处理修改学生的动作 \*/  public void handleUpdateGesture(Student stu) throws Exception;  /\*\* 处理读取文件的动作 \*/  public Object handleLoadGesture() throws Exception;  /\*\* 处理保存文件的动作 \*/  public void handleSaveGesture(List<Student> it) throws Exception;  /\*\* 处理获取全部学生的动作 \*/  public void handleGetAllStudent();    }  如例程3所示是 InfoSysupdateel接口的源程序。InfoSysupdateel接口中声明了操纵文件的一系列方法，这些方法用于添加、更新、删除和查询文件中的学生信息等。  例程3：InfoSysupdateel.java  package updateel;  import java.rmi.RemoteException;  import java.util.List;  import java.util.Set;  import InfoSys.InfoSysException;  import InfoSys.view.InfoSysView;  import InfoSys.vo.\*;  public interface InfoSysupdateel {  public void add(Student stu) throws InfoSysException,Exception;  public void show(Student stu) throws InfoSysException,Exception;  public Student findByID(String id) throws InfoSysException,Exception;    public Student findByName(String name) throws InfoSysException,Exception;  public void delete(Student stu) throws InfoSysException,Exception;  public void update(Student stu) throws InfoSysException,Exception;    public Object load() throws InfoSysException,Exception;    public void save(List<Student> it) throws InfoSysException,Exception;    public Set<Student> getAllStudent()throws InfoSysException,Exception;  /\*\*注册视图，以便当模型修改了数据库中的人员信息时可以回调视图的刷新界面方法\*/  public void addChangeListener(InfoSysView iv)throws RemoteException;  public void fireupdateelChangeEvent(Student stu);  }  Store Gui 类负责生成图形界面。如例程 4 所示是 Store Gui 类的源程  序。  例程4：StudentGui.java  package InfoSys.view;  import javax.swing.\*;  import javax.swing.table.DefaultTableupdateel;  import InfoSys.vo.Student;  import java.awt.\*;  import java.awt.event.ActionListener;  import java.util.Iterator;  import java.util.Set;  public class Gui\_stu extends JFrame {  //界面的主要窗体组件  protected JFrame frame;  protected Container contentPane;  protected CardLayout card = new CardLayout();  protected JPanel cardPan = new JPanel();    //包含各种按钮的选择面板上的组件  protected JPanel selPan = new JPanel();  protected JButton stuBt = new JButton("学生详细信息");  protected JButton allStuBt = new JButton("所有学生清单");    //显示单个学生的面板上的组件  protected JPanel stuPan=new JPanel();  protected JLabel idLb=new JLabel("ID");  protected JLabel nameLb=new JLabel("学生姓名");  protected JLabel ageLb=new JLabel("学生年龄");  protected JLabel gradeLb=new JLabel("学生成绩");    public static JTextField idTf=new JTextField(25);  public static JTextField nameTf=new JTextField(25);  public static JTextField ageTf=new JTextField(25);  public static JTextField gradeTf=new JTextField(25);    protected JButton addBt=new JButton("添加学生");  protected JButton delBt=new JButton("删除学生");  protected JButton findByIDBt=new JButton("按ID查找学生");  protected JButton findByNameBt=new JButton("按姓名关键字查找学生");  protected JButton updateBt=new JButton("修改学生信息");  protected JButton showBt=new JButton("更新信息");    //列举所有学生的面板上的组件  protected JPanel allStuPan=new JPanel();  protected JLabel allStuLb=new JLabel("所有学生清单 ",SwingConstants.CENTER);  protected JTextArea allStuTa=new JTextArea();  protected JScrollPane allStuSp=new JScrollPane(allStuTa);  //table头  String[] tableHeaders={"ID","姓名","年龄","成绩"};  JTable table;  JScrollPane tablePane;  DefaultTableupdateel tableupdateel;  /\*\* 构造方法 \*/  public Gui\_stu(){  buildDisplay();  }  //日志面板上的组件  protected JPanel logPan=new JPanel();  protected JLabel logLb=new JLabel("操作日志 ",SwingConstants.CENTER);  protected static JTextArea logTa=new JTextArea(9,50);  protected JScrollPane logSp=new JScrollPane(logTa);    /\*\* 显示单个学生面板 StuPan \*/  public void refreshStuPan(Student stu){  showCard("student");  int index = table.getSelectedRow(); //获取选中行  stu.putID(table.getValueAt(index,0).toString());  stu.putName(table.getValueAt(index,1).toString());  stu.putAge(Integer.parseInt((String)table.getValueAt(index,2)));  stu.putgrade(Float.parseFloat((String)table.getValueAt(index,3)));  if(stu==null || stu.getID().equals("-1")){  idTf.setText(null);  nameTf.setText(null);  ageTf.setText(null);  gradeTf.setText(null);  return;  }  idTf.setText(new Long(stu.getID()).toString());  nameTf.setText(stu.getName().trim());  ageTf.setText(String.valueOf(stu.getAge()));  gradeTf.setText(String.valueOf(stu.getgrade()));  }    /\*\* 显示所有客户面板 allStuPan \*/  public void refreshAllStuPan(Set stus){  showCard("allstudents");    String newData[][];  newData=new String[stus.size()][4];  Iterator it=stus.iterator();  int i=0;  while(it.hasNext()){  Student stu=(Student) it.next();  newData[i][0]=new Integer(stu.getID()).toString();  newData[i][1]=stu.getName();  newData[i][2]=String.valueOf(stu.getAge());  newData[i][3]=String.valueOf(stu.getgrade());  i++;  }  tableupdateel.setDataVector(newData,tableHeaders);  }    /\*\* 在日志面板 log Pan 中添加日志信息 \*/  public static void updateLog(String msg){  logTa.append(msg+"\n");  }    /\*\* 获得学生面板 stuPan 上用户输入的 ID \*/  public String getStuIdOnCustPan(){  try{  return idTf.getText().trim();  }catch(Exception e){  updateLog(e.getMessage());  return null;  }  }      /\*\* 显示单个客户面板 stuPan 或者所有客户面板 allStuPan \*/  void showCard(String cardStr){    card.show(cardPan,cardStr);  allStuPan.setVisible(true);  }        /\*\* 创建图形界面 \*/  private void buildDisplay(){  //frame=new JFrame("学生信息管理系统");  setTitle("学生信息管理系统");  buildSelectionPanel();  buildStuPanel();  buildAllStuPanel();  buildLogPanel();  /\*\* carPan 采用 CardLayout 布局管理器，包括 stuPan 和  allStuPan 两张卡片 \*/  cardPan.setLayout(card);  cardPan.add(stuPan,"student");  cardPan.add(allStuPan,"allstudents");  //向主窗体中加入各种面板    this.setLayout(new BorderLayout());  add(cardPan,BorderLayout.CENTER);  add(selPan,BorderLayout.NORTH);  add(logPan,BorderLayout.SOUTH);  this.pack();  this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  this.setVisible(true);    }  //获得单个人员面板上用户输入的人员信息  public Student getStudentOnCustPanel()  {  try  {  String custid=idTf.getText().trim();  String cid = custid.equals("")?"0":custid;  return new Student(cid,  nameTf.getText(),  Integer.parseInt(ageTf.getText().trim()),  Float.parseFloat(gradeTf.getText()));  }catch(Exception e)  {  updateLog(e.getMessage());  return null;  }  }  /\*\* 创建选择面板 selPan \*/  private void buildSelectionPanel(){  selPan.add(stuBt);  selPan.add(allStuBt);  }    /\*\* 为选择面板 selPan 中的两个按钮注册监听器 \*/  public void addSelectionPanelListeners(ActionListener a[]){  int len=a.length;  if(len!=2){ return;}  stuBt.addActionListener(a[0]);  allStuBt.addActionListener(a[1]);  }  public static void main(String[]args)  {  new Gui\_stu().setVisible(true);;    }  /\*\* 创建单个客户stuPan 面板 \*/  private void buildStuPanel(){  stuPan.setLayout(new GridLayout(7,2,2,2));  stuPan.add(idLb);  stuPan.add(idTf);  stuPan.add(nameLb);  stuPan.add(nameTf);  stuPan.add(ageLb);  stuPan.add(ageTf);  stuPan.add(gradeLb);  stuPan.add(gradeTf);  stuPan.add(addBt);  stuPan.add(delBt);  stuPan.add(findByIDBt);  stuPan.add(findByNameBt);  stuPan.add(updateBt);  stuPan.add(showBt);  }    /\*\* 为单个客户面板 stuPan 中的6 个按钮注册监听器 \*/  public void addStuPanelListeners(ActionListener a[]){  addBt.addActionListener(a[0]);  delBt.addActionListener(a[1]);  findByIDBt.addActionListener(a[2]);  findByNameBt.addActionListener(a[3]);  updateBt.addActionListener(a[4]);  showBt.addActionListener(a[5]);  }    /\*\* 创建所有学生 allStuPan 面板 \*/  private void buildAllStuPanel(){  allStuPan.setLayout(new BorderLayout());  allStuPan.add(allStuLb,BorderLayout.NORTH);  allStuTa.setText("all student display");  tableupdateel=new DefaultTableupdateel(tableHeaders,10);  table=new JTable(tableupdateel);  tablePane=new JScrollPane(table);  allStuPan.add(tablePane,BorderLayout.CENTER);  Dimension dim=new Dimension(500,150);  table.setPreferredScrollableViewportSize(dim); //设置此表视口的首选大小。  }    /\*\* 创建日志面板\*/  private void buildLogPanel(){  logPan.setLayout(new BorderLayout());  logSp.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_ALWAYS);  logSp.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_ALWAYS);  logPan.add(logLb,BorderLayout.NORTH);  this.logPan.add(logSp,BorderLayout.CENTER); }      }    **4 调试分析**（小三黑体，居中）  添加学生信息时，如果文件中已存在相同ID的学生，仍然会成功。  解决：运行时先读取文本信息并储存  **5 测试**（小三黑体，居中）  添加学生：在左边输入相关信息，在右边显示。  8M$V1YW`0B@FDO71LWE_(]W  根据ID查询学生：在下面相应的文本框输入ID，在右边显示。  YKOV3%LCVY66%GGZ0U(`VKT  根据姓名模糊查询：在下面输入相关的姓名，将有关联的信息在右边显示  **S4{HB3_9ITT%MEOA]7C%`PL**  **根据ID删除学生：在下面输入要删除的学生ID，点击删除，即可删除**  **2VQ14Z6X869ZOUU%SZ~H3~V**  **更新学生：就可以将全部的学生信息显示**  **]YJJU02%2E%N{BHLU72MB`Y**  **点击所有学生清单：此界面关闭，进入另外一个界面**  **再次点击所有学生清单，显示所有学生信息**  **@X5HYSP}UQF9QK8}R_C}SEG**    **心得与体会**（小三黑体，居中）  经过一学期的JAVA学习，对JAVA编程有了一定的认识，在做这个程序设计的初期，感觉自己无从下手。之后我查阅各种资料，认真分析。此次任务是编写基于MVC架构的GUI界面信息管理统，实现对学生信息的增删改查，同时利用IO数据流，对数据进行了保存和提取。当程序运行结果看到自己所想达的要求界面时感到很欣慰，虽然只是一个小的项目，但是却是自己第一次完成的一个完整项目，我也从中学到了很多书本知识不能体现的东西，对JAVA也有了更深一步的认识与了解。  通过这次的实训，我发现自己平时对于编程实际操作的匮乏，没有把书本上的知识真正的掌握牢固，仍然有许多的问题考虑不全面，同时，使我对java的面向对象有了更深一步的认识，也增加了平时上课没有学到的知识，通过老师给的资料，学习了泛型和MVC架构，，让我对java的整体又有了更深一步的学习。  MVC是一个设计模式，它强制性的使应用程序的输入、处理和输出分开。使用MVC应用程序被分成三个核心部件：模型、视图、控制器。它们各自处理自己的任务。  视图是用户看到并与之交互的界面。如何处理应用程序的界面变得越来越有挑战性。MVC一个大的好处是它能为你的应用程序处理很多不同的视图。  控制器接受用户的输入并调用模型和视图去完成用户的需求。所以当单击Web页面中的超链接和发送HTML表单时，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后用确定用哪个视图来显示模型处理返回的数据。  首先控制器接收用户的请求，并决定应该调用哪个模型来进行处理，然后模型用业务逻辑来处理用户的请求并返回数据，最后控制器用相应的视图格式化模型返回的数据，并通过表示层呈现给用户。  **参考文献** （小三黑体，居中）  [1]百度：https://www.baidu.com/  [2]CSDN：http://blog.csdn.net/  **附 录**（小三黑体，居中） |