Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**Иркутский национальный исследовательский технический университет**

|  |
| --- |
| Институт информационных технологий и анализа данных |
| наименование института |

|  |
| --- |
| **Отчет** |
| к лабораторной работе №5  по дисциплине «ООП»  **«**Разработка приложения с графическим пользовательским интерфейсом**»** |
| наименование темы  Вариант №1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | ЭВМб-19-2 |  |  |  | А. А. Бодров |
|  |  | шифр |  | подпись |  | И. О. Фамилия |
| Проверил |  |  |  |  |  | Т. В. Маланова |
|  |  | должность |  | подпись |  | И. О. Фамилия |
| Работа защищена с оценкой | | | |  | | |

Иркутск 2020 г.

# 1 Постановка задачи

*Цель работы:*

1. Получение общего представления о шаблоне программирования MVC (Model-View-Controller) и его практического применения.

2. Получение практического опыта работы с базовыми графическими библиотеками Java на примере Java Swing.

*Содержание задания:*

Необходимо разработать графический пользовательский интерфейс средствами Java для работы с классами, реализованными при выполнении лабораторной работы №4.

Графический пользовательский интерфейс должен предоставлять возможности работы со всеми возможными функциями программы (ввод и вывод данных в соответствующей форме, изменение условий выполнения задания, работа с файлами и т.д.).

Состав используемых компонентов определяется самостоятельно, но для них должны быть предусмотрены соответствующие планировщики раскладки компонентов в основных контейнерах.

Классы, реализующие графический пользовательский интерфейс, должны содержать только обращения к методам классов для выполнения заданий.

*Индивидуальное задание:*

Спроектировать и разработать приложение с графическим интерфейсом для работы с классом-группой (кафедра) в соответствии с описанной иерархией классов: студент, преподаватель, персона и завкафедрой;

# 2 Проектирование графического интерфейса

Для реализации всех необходимых функций программы, было разработана главная форма программы со строкой меню в верхней части окна.

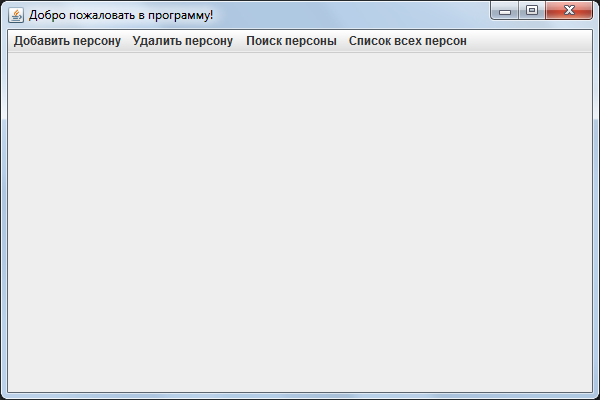
Взаимодействие с таблицей возможно через команды главного и контекстного меню программы.

Рисунок 1 – Главное окно программы

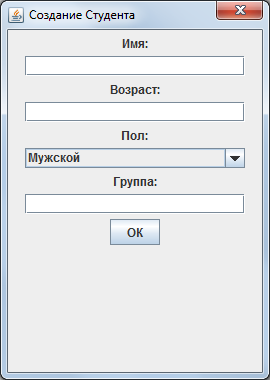
Для добавления записей таблицы было реализовано дополнительное окно ввода данных с соответствующими полями ввода информации (рис. 2).

Рисунок 2 – Окно добавления

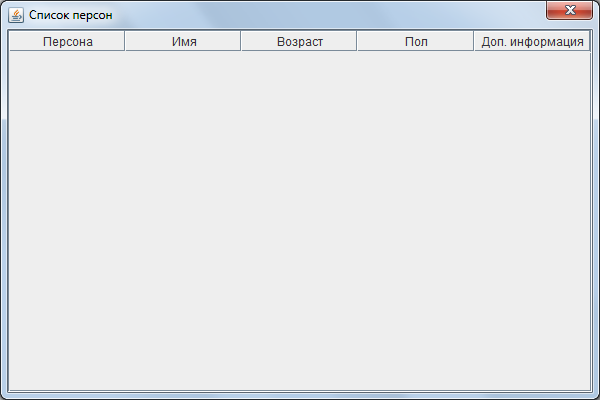
Для показа всего списка людей на экран было реализовано специальное окно (рис. 3).

Рисунок 3 - Таблица для представления данных

Кроме того в программе используется множество вспомогательных диалоговых окон для следующих целей:

получения ответа от пользователя (рис. 5);

ввода вспомогательных данных (рис. 6);

вывода справочной информации (рис. 7 и 8);

вывода предупреждений (рис. 9);

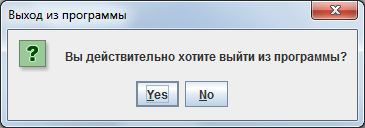
Рисунок 5 – Подтверждающее диалоговое окно

Рисунок 6 – Диалоговое окно ввода данных

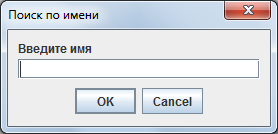
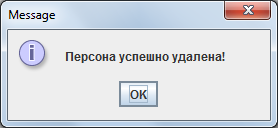


Рисунок 7 – Информационное диалоговое окно

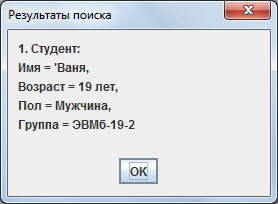
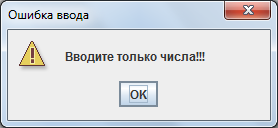
 Рисунок 8 - Диалоговое окно вывода данных

Рисунок 9 – Предупреждающее диалоговое окно

# 3 Описание структуры классов

Весь программы код разделен на три основных части: модель данных (*Model*), интерфейс взаимодействия с пользователем (*View*) и контроллера управления программой (*Controll*).

Все классы находятся в основном пакете *com.bodrov* и разбиты на подпакеты в зависимости от выполняемых ими функций. В основном пакете находится главный класс программы *Program*, имеющий единственный метод *main*, являющийся точкой входа в программу.

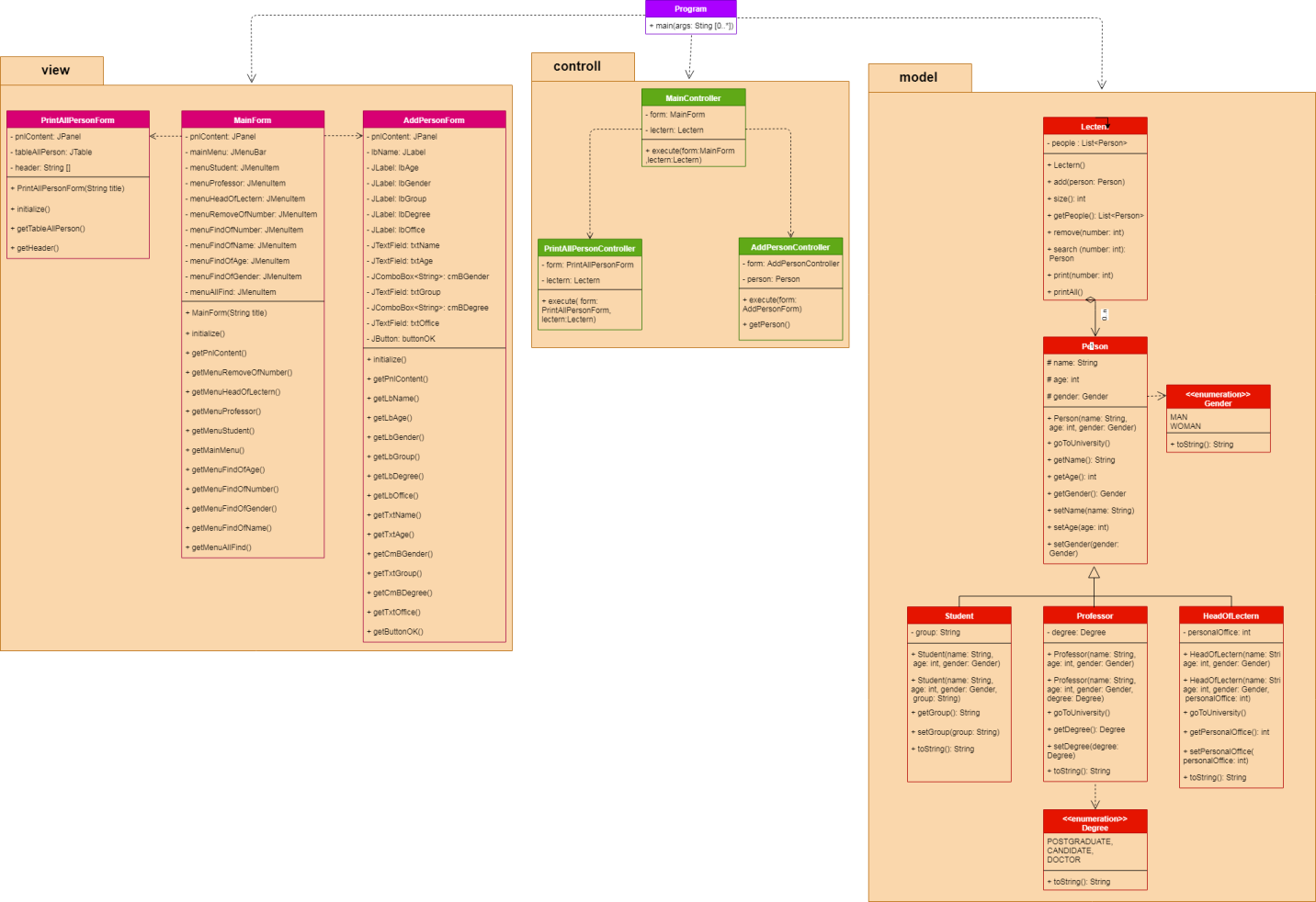
Общая структура классов представлена в виде UML-диаграммы проекта и изображена на рис.10.

Рисунок 10 – UML-диаграмма проекта

# 4 Проектирование тестов

В ходе разработки было проведено тестирование программы, результаты которого представлены в табл. 1.

|  |  |  |
| --- | --- | --- |
| № | Действие | Результат действия |
| 1 | Первоначальный запуск программы | При старте программы появляется главная форма с пустой таблицей и строкой меню в верхней части окна (рис. 11). |
| 2 | Создание новой персоны | Для того чтобы добавить новую запись, нажимаем кнопку меню *Добавить персону -> Студент* (рис. 12).  Открывается новое диалоговое окно добавления (рис. 13).  Заполняем данными поля ввода и нажимаем кнопку *OK* (рис. 14). |
| 3 | Удаление персоны | Для того чтобы удалить персону, выбираем нужную строку и нажимаем кнопку меню *Удалить персону –> Удалить по номеру* (рис. 15).  После чего вводим номер удаляемой персоны и удаляем ее (рис. 16) |
| 4 | Поиск персоны | Для того чтобы провести поиск, мы выбираем *Поиск персоны*. Далее у нас появляется выбор поиска (рис. 17).  Выбираем нужный нам поиск, а затем на экране появляется окно ввода (рис. 18)  На экране появляется окно с параметрами персоны (рис. 19) |
| 5 | Список всех персон | Чтобы получить список всех персон нужно выбрать *Список всех персон ->Весь список*  (рис. 20)  Далее появляется окно с таблицей всех наших персон (рис 21). |
| 6 | Завершение работы программы | Для закрытия программы нажимаем *на крестик главной формы* (рис. 22).  После чего подтверждаем закрытие программы нажатием кнопки *OK* (рис. 23). |

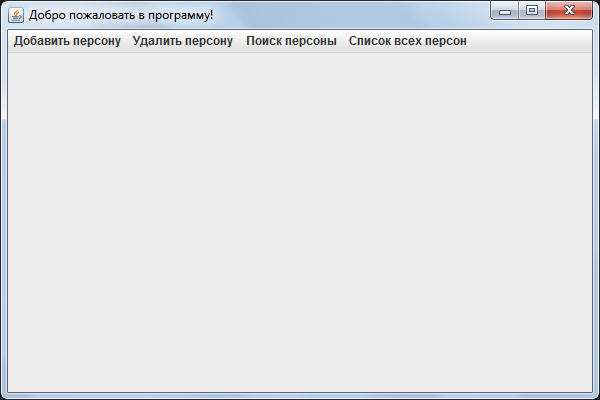
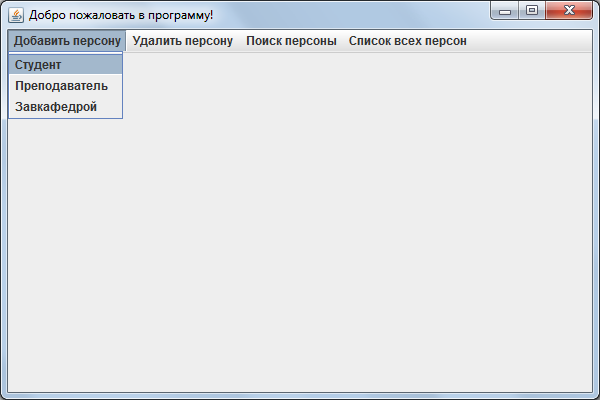
На рис. 11 изображено главное окно программы, появляющееся при запуске программы. Весь функционал вынесен в виде строки меню в верхней части окна.

Рисунок 12 – Первоначальный запуск программы

 Рисунок 12 – Создание новой персоны

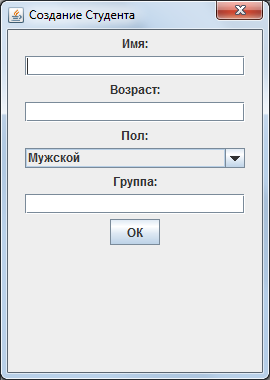
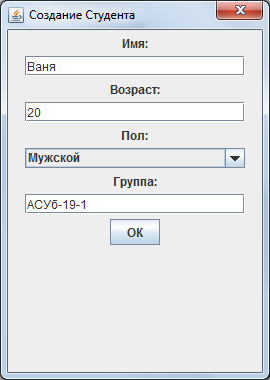
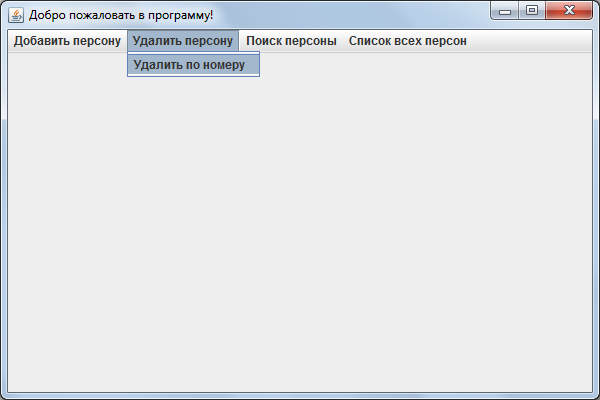
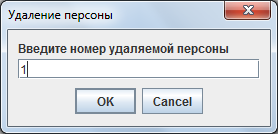
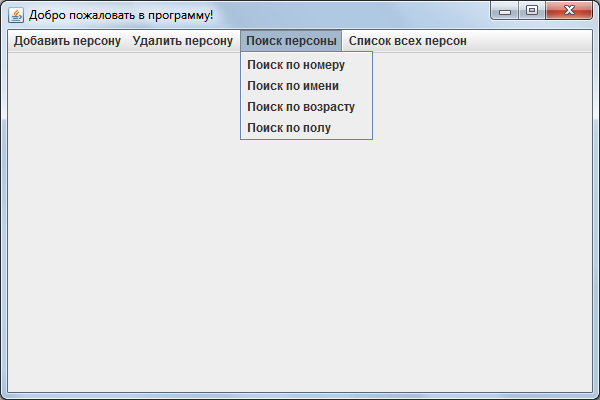
 Рисунок 13 – Открытие диалогового окна добавления записей

Рисунок 14 – Заполнение данными полей ввода

Рисунок 15 – Удаление персоны

Рисунок 16 – Вводим удаляемой номер персоны

Рисунок 17 – Поиск персоны

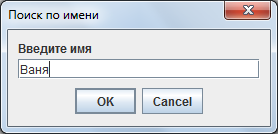
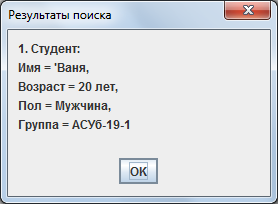
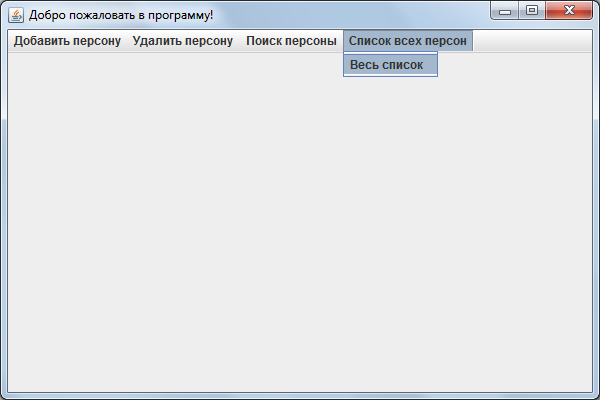
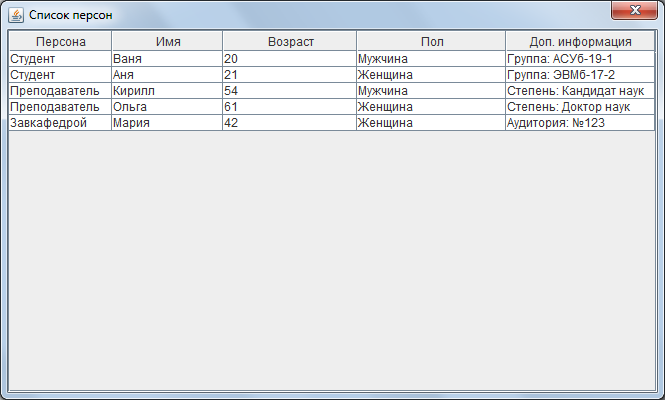
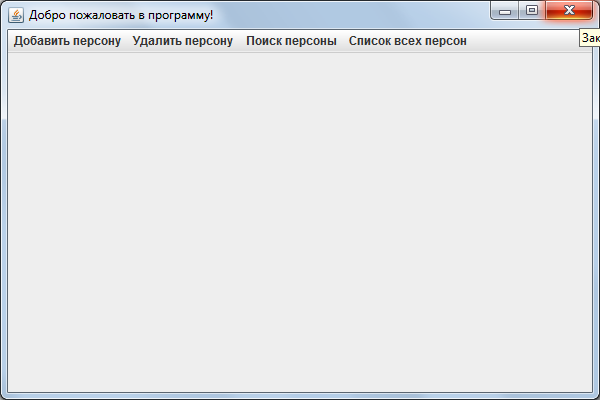
Рисунок 18 – Окно ввода для поиска персоны

Рисунок 19 – Окно с параметрами персоны

 Рисунок 20 – Выбор кнопки “Список всех персон”

Рисунок 21 – Весь список персон

Рисунок 22 – Завершение работы программы

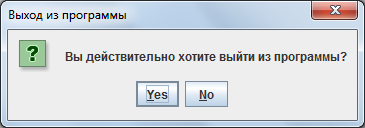


Рисунок 23 – Подтверждение выхода

# 5 Листинг программы на Java

**Главный класс Program**

package com.bodrov;  
  
import com.bodrov.controll.MainController;  
import com.bodrov.model.Lectern;  
import com.bodrov.view.MainForm;  
  
import javax.swing.\*;  
  
public class Program {  
 public static void main(String[] args) {  
 //Запуск контролера в новом потоке и передача ему интерфейса  
 SwingUtilities.*invokeLater*(() -> {  
 MainController controller = new MainController();  
 MainForm form = new MainForm("Добро пожаловать в программу!");  
 Lectern lectern = new Lectern();  
  
 controller.execute(form, lectern);  
 });  
 }  
}

**Класс MainForm**

package com.bodrov.view;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class MainForm extends JFrame {  
 private JPanel pnlContent;  
 private JMenuBar mainMenu;  
 private JMenuItem menuStudent;  
 private JMenuItem menuProfessor;  
 private JMenuItem menuHeadOfLectern;  
 private JMenuItem menuRemoveOfNumber;  
 private JMenuItem menuFindOfNumber;  
 private JMenuItem menuFindOfName;  
 private JMenuItem menuFindOfAge;  
 private JMenuItem menuFindOfGender;  
 private JMenuItem menuAllFind;  
  
  
 public MainForm(String title) {  
 initialize();// Инициализируем компоненты  
  
 setTitle(title);//Задаем заголовок окна  
 setBounds(400, 300, 600, 400);// Задаем размеры и координамы  
 setDefaultCloseOperation(*DO\_NOTHING\_ON\_CLOSE*);// Ничего не делать при закрытии  
 setContentPane(pnlContent);//Установить панель в качестве главной  
 setVisible(true);//Сделать окно видимым  
 }  
  
 private void initialize() {  
 pnlContent = new JPanel(new BorderLayout());// Пограничное расположение(Слева направо)  
  
 //  
 mainMenu = new JMenuBar();  
  
  
 //Добавление персоны  
 JMenu menuAdd = new JMenu("Добавить персону");  
 menuStudent = new JMenuItem("Студент");  
 menuProfessor = new JMenuItem("Преподаватель");  
 menuHeadOfLectern = new JMenuItem("Завкафедрой");  
  
 //Добавление кнопок в меню (Добавление)  
 menuAdd.add(menuStudent);  
 menuAdd.add(menuProfessor);  
 menuAdd.add(menuHeadOfLectern);  
  
  
  
 //Удалить персону  
 JMenu menuRemove = new JMenu("Удалить персону");  
 menuRemoveOfNumber = new JMenuItem("Удалить по номеру");  
  
 //Дабавление кнопки в меню (Удаление)  
 menuRemove.add(menuRemoveOfNumber);  
  
  
 //Поиск персоны  
 JMenu menuFind = new JMenu("Поиск персоны");  
 menuFindOfNumber = new JMenuItem("Поиск по номеру ");  
 menuFindOfName = new JMenuItem("Поиск по имени ");  
 menuFindOfAge = new JMenuItem("Поиск по возрасту ");  
 menuFindOfGender = new JMenuItem("Поиск по полу ");  
  
 //Добавление кнопки в меню (Поиск персоны)  
 menuFind.add(menuFindOfNumber);  
 menuFind.add(menuFindOfName);  
 menuFind.add(menuFindOfAge);  
 menuFind.add(menuFindOfGender);  
  
  
 //Вывести список всех персон  
 JMenu menuAllPerson = new JMenu("Список всех персон");  
 menuAllFind = new JMenuItem("Весь список");  
  
 //Добавление кнопки в меню (Поиск всех персон)  
 menuAllPerson.add(menuAllFind);  
  
  
  
  
 //Добавление меню в основной Bar  
 mainMenu.add(menuAdd);  
 mainMenu.add(menuRemove);  
 mainMenu.add(menuFind);  
 mainMenu.add(menuAllPerson);  
  
  
 pnlContent.add(mainMenu, BorderLayout.*NORTH*);  
 }  
  
 public JPanel getPnlContent() { return pnlContent; }  
 public JMenuItem getMenuRemoveOfNumber() { return menuRemoveOfNumber; }  
 public JMenuItem getMenuHeadOfLectern() { return menuHeadOfLectern; }  
 public JMenuItem getMenuProfessor() { return menuProfessor; }  
 public JMenuItem getMenuStudent() { return menuStudent; }  
 public JMenuBar getMainMenu() { return mainMenu; }  
 public JMenuItem getMenuFindOfAge() { return menuFindOfAge; }  
 public JMenuItem getMenuFindOfNumber() { return menuFindOfNumber; }  
 public JMenuItem getMenuFindOfGender() { return menuFindOfGender; }  
 public JMenuItem getMenuFindOfName() { return menuFindOfName; }  
 public JMenuItem getMenuAllFind() { return menuAllFind; }  
}

**Класс AddPersonForm**

package com.bodrov.view;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class AddPersonForm extends JDialog {  
  
 private JPanel pnlContent;  
 private JLabel lbName;  
 private JLabel lbAge;  
 private JLabel lbGender;  
 private JLabel lbGroup;  
 private JLabel lbDegree;  
 private JLabel lbOffice;  
  
 private JTextField txtName;  
 private JTextField txtAge;  
 private JComboBox<String> cmBGender;  
 private JTextField txtGroup;  
 private JComboBox<String> cmBDegree;  
 private JTextField txtOffice;  
  
 private JButton buttonOK;  
  
 public AddPersonForm(String title) {  
 initialize();  
  
 setTitle(title);  
 setBounds(400, 300, 270, 380);  
 setDefaultCloseOperation(*DISPOSE\_ON\_CLOSE*);  
 setMinimumSize(getSize());//Текущий размер будет минимальным  
 setContentPane(pnlContent);  
 setModal(true);  
 }  
  
 private void initialize() {  
 pnlContent = new JPanel(new FlowLayout(FlowLayout.*CENTER*,10,5));  
 lbName = new JLabel("Имя:");  
 lbAge = new JLabel("Возраст:");  
 lbGender = new JLabel("Пол:");  
 lbGroup = new JLabel("Группа:");  
 lbDegree = new JLabel("Степень:");  
 lbOffice = new JLabel("Аудитория:");  
 txtName = new JTextField();  
 txtAge = new JTextField();  
 cmBGender = new JComboBox<>(new String[]{ "Мужской", "Женский" });  
 txtGroup = new JTextField();  
 cmBDegree = new JComboBox<>(new String[]{ "Аспирант","Кандидат наук","Доктор наук" });  
 txtOffice = new JTextField();  
 buttonOK = new JButton("ОК");  
  
 //Размер текстовых полей  
 Dimension size = new Dimension(220, 20);  
 txtName.setPreferredSize(size);  
 txtAge.setPreferredSize(size);  
 txtGroup.setPreferredSize(size);  
 txtOffice.setPreferredSize(size);  
  
 //Размер ComboBox(Выпадающие списки)  
 cmBGender.setPreferredSize(size);  
 cmBDegree.setPreferredSize(size);  
  
 //Добавление всех компанентов на панельку  
 pnlContent.add(lbName);  
 pnlContent.add(txtName);  
 pnlContent.add(lbAge);  
 pnlContent.add(txtAge);  
 pnlContent.add(lbGender);  
 pnlContent.add(cmBGender);  
 pnlContent.add(lbGroup);  
 pnlContent.add(txtGroup);  
 pnlContent.add(lbDegree);  
 pnlContent.add(cmBDegree);  
 pnlContent.add(lbOffice);  
 pnlContent.add(txtOffice);  
 pnlContent.add(buttonOK);  
 }  
  
  
  
  
 public JPanel getPnlContent() { return pnlContent; }  
 public JLabel getLbName() { return lbName; }  
 public JLabel getLbAge() { return lbAge; }  
 public JLabel getLbGender() { return lbGender; }  
 public JLabel getLbGroup() { return lbGroup; }  
 public JLabel getLbDegree() { return lbDegree; }  
 public JLabel getLbOffice() { return lbOffice; }  
  
 public JTextField getTxtName() { return txtName; }  
 public JTextField getTxtAge() { return txtAge; }  
 public JComboBox<String> getCmBGender() { return cmBGender; }  
 public JTextField getTxtGroup() { return txtGroup; }  
 public JComboBox<String> getCmBDegree() { return cmBDegree; }  
 public JTextField getTxtOffice() { return txtOffice; }  
  
 public JButton getButtonOK() { return buttonOK; }  
}

**Класс PrintAllPersonForm**

package com.bodrov.view;  
  
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.\*;  
  
public class PrintAllPersonForm extends JDialog {  
 private JPanel pnlContent;  
 private JTable tableAllPerson;  
 private String [] header;  
  
 public PrintAllPersonForm(String title) {  
 initialize();// Инициализируем компоненты  
  
 setTitle(title);//Задаем заголовок окна  
 setBounds(400, 300, 600, 400);// Задаем размеры и координамы  
 setDefaultCloseOperation(*DISPOSE\_ON\_CLOSE*);// Закрытие по умолчанию  
 setContentPane(pnlContent);//Установить панель в качестве главной  
 setModal(true);  
 }  
  
 private void initialize() {  
 pnlContent = new JPanel(new BorderLayout());  
  
 header = new String[] {  
 "Персона", "Имя", "Возраст", "Пол", "Доп. информация",  
 };  
  
 String [][] data = new String[0][];  
  
 tableAllPerson = new JTable(data, header);  
 JScrollPane scrollPane = new JScrollPane(tableAllPerson);  
 pnlContent.add(scrollPane, BorderLayout.*CENTER*);  
  
 }  
  
 public JTable getTableAllPerson() {return tableAllPerson; }  
 public String[] getHeader() { return header; }  
}

**Класс Lectern**

package com.bodrov.model;  
  
import com.bodrov.model.enums.Gender;  
import com.bodrov.model.persons.Person;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class Lectern {  
 private final List<Person> people;  
  
 //Создает новую коллекцию  
 public Lectern() {  
 people = new ArrayList<>();  
 }  
  
 //Добавляет новую персону  
 public void add(Person person) {  
 people.add(person);  
 }  
 public int size() {  
 return people.size();  
 }  
  
 //Возвращаем колекцию людей  
 public List<Person> getPeople() {  
 return people;  
 }  
  
 //Удаление по номеру  
 public void remove(int number){  
 if (number > 0 && number <= people.size())  
 people.remove(number - 1);  
 }  
  
 //Поиск по номеру  
 public Person get(int number){  
 if (number > 0 && number <= people.size())  
 return people.get(number - 1);  
 return null;  
 }  
  
 //Поиск по имени  
 public List<Person> search(String name) {  
 List<Person> list = new ArrayList<>();  
 for (Person person : people) {  
 if (person.getName().equalsIgnoreCase(name))  
 list.add(person);  
 }return list;  
 }  
 public List<Person> search(int age) {  
 List<Person> list = new ArrayList<>();  
 for (Person person : people) {  
 if (person.getAge() == age)  
 list.add(person);  
 }return list;  
 }  
  
 public List<Person> search(Gender gender) {  
 List<Person> list = new ArrayList<>();  
 for (Person person : people) {  
 if (person.getGender() == gender)  
 list.add(person);  
 }return list;  
 }  
}

**Класс HeadOfLectern**

package com.bodrov.model.persons;  
  
import com.bodrov.model.enums.Gender;  
  
public class HeadOfLectern extends Person {  
 private int personalOffice;  
  
 public HeadOfLectern(String name, int age, Gender gender) {  
 super(name, age, gender);  
 }  
  
 public HeadOfLectern(String name, int age, Gender gender, int personalOffice) {  
 super(name, age, gender);  
 this.personalOffice = personalOffice;  
 }  
  
 @Override  
 public void goToUniversity() {  
 System.*out*.println("Завкафедрой следит за учебным процессом");  
 }  
  
 public int getPersonalOffice() {  
 return personalOffice;  
 }  
  
 public void setPersonalOffice(int personalOffice) {  
 this.personalOffice = personalOffice;  
 }  
  
 @Override public String toString() {  
 return "Зав. кафедрой: \n\tИмя = '" + name + "," +  
 "\n\tВозраст = " + age + " лет," +  
 "\n\tПол = " + gender.getName() + ", " +  
 "\n\tАудитория = №" + personalOffice + "\n\n";  
 }  
}

**Класс Person**

package com.bodrov.model.persons;  
  
import com.bodrov.model.enums.Gender;  
  
public abstract class Person {  
 protected String name;  
 protected int age;  
 protected Gender gender;  
  
 //Конструктор передачи параметров  
 public Person(String name, int age, Gender gender) {  
 this.name = name;  
 this.age = age;  
 this.gender = gender;  
 }  
  
 public abstract void goToUniversity();  
  
 public String getName() {  
 return name;  
 }  
  
 public int getAge() {  
 return age;  
 }  
  
 public Gender getGender() {  
 return gender;  
 }  
  
  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setAge(int age) {  
 this.age = age;  
 }  
  
 public void setGender(Gender gender) {  
 this.gender = gender;  
 }  
}

**Класс Professor**

package com.bodrov.model.persons;  
  
import com.bodrov.model.enums.Degree;  
import com.bodrov.model.enums.Gender;  
  
public class Professor extends Person {  
 private Degree degree;  
  
 public Professor(String name, int age, Gender gender) {  
 super(name, age, gender);  
 }  
  
 public Professor(String name, int age, Gender gender, Degree degree) {  
 super(name, age, gender);  
 this.degree = degree;  
 }  
  
 @Override  
 public void goToUniversity() {  
 System.*out*.println("Преподаватель ведет лекции и учит студентов");  
 }  
  
 public Degree getDegree() {  
 return degree;  
 }  
  
 public void setDegree(Degree degree) {  
 //Проверка( Не равна null)  
 if (degree != null)  
 this.degree = degree;  
 }  
  
 @Override public String toString() {  
 return "Преподаватель: \n\tИмя = '" + name + "," +  
 "\n\tВозраст = " + age + " лет," +  
 "\n\tПол = " + gender.getName() + ", " +  
 "\n\tНаучная степень = " + degree.getName() + "\n\n";  
 }  
}

**Класс Student**

package com.bodrov.model.persons;  
  
import com.bodrov.model.enums.Gender;  
  
public class Student extends Person {  
 private String group;  
 //Вызов конструктора в родительском классе  
 public Student(String name, int age, Gender gender) {  
 super(name, age, gender);  
  
 }  
 //Создание нового студента  
 public Student(String name, int age, Gender gender, String group) {  
 super(name, age, gender);  
 this.group = group;  
 }  
  
 @Override//Метод переопределяется(Другая реализация)  
 public void goToUniversity() {  
 System.*out*.println("Студент учится и получает знания");  
 }  
  
 public String getGroup() {  
 return group;  
 }  
  
 public void setGroup(String group) {  
 //Проверка( Не равна null и не пустая)  
 if (group != null && !group.isEmpty())  
 this.group = group;  
 }  
  
 @Override public String toString() {  
 return "Студент: \n\tИмя = '" + name + "," +  
 "\n\tВозраст = " + age + " лет," +  
 "\n\tПол = " + gender.getName() + ", " +  
 "\n\tГруппа = " + group + "\n\n";  
 }  
}

**Перечисление Degree**

package com.bodrov.model.enums;  
  
public enum Degree {  
 *POSTGRADUATE*("Аспирант"),  
 *CANDIDATE*("Кандидат наук"),  
 *DOCTOR*("Доктор наук");  
  
 private final String name;  
 Degree(String name) { this.name = name; }  
 public String getName() { return name; }  
  
 public static Degree convert(int index) {  
 for (Degree degree : *values*()) {  
 if (degree.ordinal() == index)  
 return degree;  
 } return null;  
 }  
}

**Перечисление Gender**

package com.bodrov.model.enums;  
  
public enum Gender {  
 *MAN*("Мужчина"),  
 *WOMAN*("Женщина");  
  
 private final String name;  
 Gender(String name) { this.name = name; }  
 public String getName() { return name; }  
  
 public static Gender convert(int index) {  
 for (Gender gender : *values*()) {  
 if (gender.ordinal() == index)  
 return gender;  
 } return null;  
 }  
}

**Класс AddPersonController**

package com.bodrov.controll;  
  
import com.bodrov.model.enums.\*;  
import com.bodrov.model.persons.\*;  
import com.bodrov.view.AddPersonForm;  
  
import javax.swing.\*;  
import java.awt.event.KeyAdapter;  
import java.awt.event.KeyEvent;  
  
public class AddPersonController {  
 private AddPersonForm form;  
 private Person person;  
  
 public void execute (AddPersonForm form){  
 this.form = form;  
  
 this.form.getButtonOK().addActionListener(e -> {  
 //Получение данных с компонентов формы  
 String name = form.getTxtName().getText();  
 Gender gender = Gender.*convert*(form.getCmBGender().getSelectedIndex());  
 int age = Integer.*parseInt*(form.getTxtAge().getText());  
  
  
 //по видимости компонента определяем нажатую кнопку  
 if (form.getTxtGroup().isVisible()) {  
 String group = form.getTxtGroup().getText();  
 person = new Student(name, age, gender, group);  
  
  
 } else if (form.getTxtOffice().isVisible()) {  
 int office = Integer.*parseInt*(form.getTxtOffice().getText());  
 person = new HeadOfLectern(name,age,gender, office);  
  
  
 } else if (form.getCmBDegree().isVisible()) {  
 Degree degree = Degree.*convert*(form.getCmBDegree().getSelectedIndex());  
 person = new Professor(name, age, gender, degree);  
 }  
 form.dispose();  
 });  
  
 //Обработчик ввода символов (Возраст)  
 this.form.getTxtAge().addKeyListener(new KeyAdapter() {  
 @Override  
 public void keyTyped(KeyEvent e) {  
 //Берем у события источник, который является текстовым полем и берем у него введенный текст  
 String text = ((JTextField) e.getSource()).getText();  
  
 //Если введеный символ не является цифрой  
 if (!Character.*isDigit*(e.getKeyChar())  
 || text.length() >= 2  
 || text.length() == 0 && e.getKeyChar() == '0')  
 e.consume();//Остановить событие  
 }  
 });  
  
 //Обработчик ввода символов (Имя)  
 this.form.getTxtName().addKeyListener((new KeyAdapter() {  
 @Override  
 public void keyTyped(KeyEvent e) {  
 String text = ((JTextField) e.getSource()).getText();  
  
 if (e.getKeyChar() < 'А' || e.getKeyChar() > 'я'  
 || text.length() == 0 && Character.*isLowerCase*(e.getKeyChar())  
 || text.length() != 0 && Character.*isUpperCase*(e.getKeyChar()))  
  
 e.consume();//Остановить событие  
  
 }  
 }));  
  
  
 form.setVisible(true);  
 }  
  
 public Person getPerson() { return person; }  
}

**Класс MainController**

package com.bodrov.controll;  
  
import com.bodrov.model.Lectern;  
import com.bodrov.model.enums.Gender;  
import com.bodrov.model.persons.Person;  
import com.bodrov.view.\*;  
  
import javax.swing.\*;  
import java.awt.event.WindowAdapter;  
import java.awt.event.WindowEvent;  
import java.util.List;  
  
public class MainController {  
 private MainForm form;  
 private Lectern lectern;  
  
 //Вызов обработчиков событий  
 public void execute(MainForm form, Lectern lectern) {  
 this.form = form;  
 this.lectern = lectern;  
  
  
 //УДАЛЕНИЕ  
 this.form.getMenuRemoveOfNumber().addActionListener(e -> {  
 //Вызов дилогового окна ввода  
 //Перевод сткроки в число и сохраняем  
  
 try {  
 int number = Integer.*parseInt*(JOptionPane.*showInputDialog*(form,  
 "Введите номер удаляемой персоны",  
 "Удаление персоны", JOptionPane.*PLAIN\_MESSAGE*));  
  
 if (lectern.size() > 0) {  
  
 if (number > 0 && number <= lectern.size()) {  
 lectern.remove(number);  
 JOptionPane.*showMessageDialog*(form,  
 "Персона успешно удалена!");  
 } else JOptionPane.*showMessageDialog*(form,  
 "Персона с таким номером отсутствует!",  
 "Ошибка удаления", JOptionPane.*ERROR\_MESSAGE*);  
  
 } else JOptionPane.*showMessageDialog*(form,  
 "Список пуст!");  
  
 } catch (NumberFormatException exception) {  
 JOptionPane.*showMessageDialog*(form,  
 "Вводите только числа!!!",  
 "Ошибка ввода", JOptionPane.*WARNING\_MESSAGE*);  
 }  
 });  
  
  
  
 //CПИСОК ПЕРСОН  
 this.form.getMenuAllFind().addActionListener(e -> {  
 PrintAllPersonController controller = new PrintAllPersonController();  
 PrintAllPersonForm dialog = new PrintAllPersonForm("Список персон");  
 controller.execute(dialog, lectern);  
 });  
  
  
  
  
 //ВЫХОД ИЗ ПРОГРАММЫ  
 this.form.addWindowListener(new WindowAdapter() {  
 @Override  
 public void windowClosing(WindowEvent e) {  
 //Будет хранить нажатую кнопку пользователя  
 int option = JOptionPane.*showConfirmDialog*(form,  
 "Вы действительно хотите выйти из программы?",  
 "Выход из программы",JOptionPane.*YES\_NO\_OPTION*);  
  
 if (option == JOptionPane.*YES\_OPTION*)  
 form.dispose();  
  
  
 }  
 });  
  
  
  
  
 //ДОБАВЛЕНИЕ  
 //Студент  
 this.form.getMenuStudent().addActionListener(e -> {  
 //Создали контролер, который обрабатывает события  
 AddPersonController controller = new AddPersonController();  
 AddPersonForm addPersonForm = new AddPersonForm("Создание Студента");  
  
 addPersonForm.getLbOffice().setVisible(false);  
 addPersonForm.getLbDegree().setVisible(false);  
 addPersonForm.getTxtOffice().setVisible(false);  
 addPersonForm.getCmBDegree().setVisible(false);  
  
 //Вызов контроллера  
 controller.execute(addPersonForm);  
 //Добаление персоны в кафедру  
 lectern.add(controller.getPerson());  
  
 });  
  
  
 //Профессор  
 this.form.getMenuProfessor().addActionListener(e -> {  
  
 AddPersonController controller = new AddPersonController();  
 AddPersonForm addPersonForm = new AddPersonForm("Создание Профессора");  
  
 addPersonForm.getLbGroup().setVisible(false);  
 addPersonForm.getLbOffice().setVisible(false);  
 addPersonForm.getTxtGroup().setVisible(false);  
 addPersonForm.getTxtOffice().setVisible(false);  
  
 controller.execute(addPersonForm);  
 lectern.add(controller.getPerson());  
  
  
 });  
  
 //Завкафедрой  
 this.form.getMenuHeadOfLectern().addActionListener(e -> {  
  
 AddPersonController controller = new AddPersonController();  
 AddPersonForm addPersonForm = new AddPersonForm("Создание Завкафедрой");  
  
 addPersonForm.getLbGroup().setVisible(false);  
 addPersonForm.getLbDegree().setVisible(false);  
 addPersonForm.getTxtGroup().setVisible(false);  
 addPersonForm.getCmBDegree().setVisible(false);  
  
 controller.execute(addPersonForm);  
 lectern.add(controller.getPerson());  
  
  
 });  
  
  
  
  
 //ПОИСК ПО ПАРАМЕТРАМ  
 //Номеру  
 this.form.getMenuFindOfNumber().addActionListener(e -> {  
  
 if (lectern.size() > 0) {  
 try {  
 int number = Integer.*parseInt*(JOptionPane.*showInputDialog*(form,  
 "Введите номер", "Поиск по номеру", JOptionPane.*PLAIN\_MESSAGE*));  
 Person person = lectern.get(number);  
  
 if (person != null) {  
 JOptionPane.*showMessageDialog*(form, person,  
 "Результаты поиска", JOptionPane.*PLAIN\_MESSAGE*);  
 } else JOptionPane.*showMessageDialog*(form,  
 "Ничего не найдено!","Ошибка!" ,  
 JOptionPane.*WARNING\_MESSAGE*);  
  
 } catch (NumberFormatException ignored) { }  
  
 } else JOptionPane.*showMessageDialog*(form,  
 "Список пуст!");  
 });  
  
 //Имя  
 this.form.getMenuFindOfName().addActionListener(e -> {  
  
 if (lectern.size() > 0) {  
 String name = JOptionPane.*showInputDialog*(form,  
 "Введите имя", "Поиск по имени", JOptionPane.*PLAIN\_MESSAGE*);  
 List<Person> people = lectern.search(name);  
 String [] strings = new String[people.size()];  
  
 for (int i = 0; i < people.size(); i++) {  
 strings[i] = (i+1) + ". " + people.get(i).toString();  
 }  
  
 if (people.size() != 0) {  
 JOptionPane.*showMessageDialog*(form, strings,  
 "Результаты поиска", JOptionPane.*PLAIN\_MESSAGE*);  
 } else JOptionPane.*showMessageDialog*(form, "Ничего не найдено!","Ошибка!" , JOptionPane.*WARNING\_MESSAGE*);  
 } else JOptionPane.*showMessageDialog*(form,  
 "Список пуст!");  
 });  
  
 //Возраст  
 this.form.getMenuFindOfAge().addActionListener(e -> {  
  
 if (lectern.size() > 0) {  
 try {  
 int age = Integer.*parseInt*(JOptionPane.*showInputDialog*(form,  
 "Введите возраст", "Поиск по возрасту", JOptionPane.*PLAIN\_MESSAGE*));  
 List<Person> people = lectern.search(age);  
 String[] strings = new String[people.size()];  
  
 for (int i = 0; i < people.size(); i++) {  
 strings[i] = (i + 1) + ". " + people.get(i).toString();  
 }  
  
 if (people.size() != 0) {  
 JOptionPane.*showMessageDialog*(form, strings,  
 "Результаты поиска", JOptionPane.*PLAIN\_MESSAGE*);  
 } else  
 JOptionPane.*showMessageDialog*(form,  
 "Ничего не найдено!", "Ошибка!",  
 JOptionPane.*WARNING\_MESSAGE*);  
  
 } catch (NumberFormatException ignored) { }  
  
 } else JOptionPane.*showMessageDialog*(form,  
 "Список пуст!");  
 });  
  
 //Полу  
 this.form.getMenuFindOfGender().addActionListener(e -> {  
  
 if (lectern.size() > 0) {  
  
 //Вспомогательный массив для определния пола  
 String [] genders = new String [] { "Мужской", "Женский" };  
  
 //Приведение к строке выбранного значения из выпадающего списка  
 String result = (String) JOptionPane.*showInputDialog*(form,  
 "Выберите пол", "Поиск по полу", JOptionPane.*PLAIN\_MESSAGE*,  
 null, genders, genders[0]);  
  
 //Приведение строки к классу Gender  
 if (result != null) {  
 Gender gender = result.equals(genders[0]) ? Gender.*MAN* : Gender.*WOMAN*;  
  
 List<Person> people = lectern.search(gender);  
 String[] strings = new String[people.size()];  
  
 for (int i = 0; i < people.size(); i++) {  
 strings[i] = (i + 1) + ". " + people.get(i).toString();  
 }  
  
 if (people.size() != 0) {  
 JOptionPane.*showMessageDialog*(form, strings,  
 "Результаты поиска", JOptionPane.*PLAIN\_MESSAGE*);  
  
 } else JOptionPane.*showMessageDialog*(form,  
 "Ничего не найдено!", "Ошибка!",  
 JOptionPane.*WARNING\_MESSAGE*);  
 }  
 } else JOptionPane.*showMessageDialog*(form, "Список пуст!");  
 });  
 }  
}

**Класс PrintAllPersonController**

package com.bodrov.controll;  
  
import com.bodrov.model.Lectern;  
import com.bodrov.model.persons.HeadOfLectern;  
import com.bodrov.model.persons.Person;  
import com.bodrov.model.persons.Professor;  
import com.bodrov.model.persons.Student;  
import com.bodrov.view.PrintAllPersonForm;  
  
import javax.swing.table.DefaultTableModel;  
import java.awt.event.FocusAdapter;  
import java.awt.event.FocusEvent;  
  
  
public class PrintAllPersonController {  
 private PrintAllPersonForm form;  
 private Lectern lectern;  
  
 public void execute(PrintAllPersonForm form, Lectern lectern) {  
 this.form = form;  
 this.lectern = lectern;  
  
  
 form.getTableAllPerson().addFocusListener(new FocusAdapter() {  
 @Override  
 public void focusGained(FocusEvent e) {  
 String[][] data = new String[lectern.size()][];  
  
 for (int i=0; i< lectern.size();i++) {  
 Person person = lectern.get(i+1);  
 String nameClass = "";  
 String addInfo = "";  
  
 //Приведение к определенному классу и получения доп. информации  
 if (person instanceof Student) {  
 nameClass = "Студент";  
 addInfo = "Группа: " + ((Student) person).getGroup();  
 }else if (person instanceof Professor) {  
 nameClass = "Преподаватель";  
 addInfo = "Степень: " + ((Professor) person).getDegree().getName();  
 }else if (person instanceof HeadOfLectern){  
 nameClass = "Завкафедрой";  
 addInfo = "Аудитория: №" + ((HeadOfLectern) person).getPersonalOffice();  
 }  
  
 //Константный массив  
 data[i] = new String[] {  
 nameClass,  
 person.getName(),  
 String.*valueOf*(person.getAge()),  
 person.getGender().getName(),  
 addInfo  
 };  
 }  
 //У таблицы обновляем ее модель  
 form.getTableAllPerson().setModel(new DefaultTableModel(data, form.getHeader()));  
 }  
 });  
  
 form.setVisible(true);//Сделать окно видимым  
 }  
}

# Список литературы

1. API Spec**if**ication **for** the Java 7 SE. [официальный сайт] URL:

<http://docs.oracle.com/javase/7/docs/api> (дата обращения: 10.11.2020)

2. The Java Tutorials. SE [электронный ресурс], URL:

<http://docs.oracle.com/javase/tutorial/index.html> (дата обращения: 12.11.2020)

3. Васильев А. Н. Java. Объектно-ориентированное программирование: для магистров и бакалавров. Базовый курс по объектно-ориентированному программированию / А. Н. Васильев. – СПб: Питер, 2012. – 395 с.

4. Отношения классов — от UML к коду [Электронный ресурс] // Хабра-хабр: [сайт]. [2006]. URL: <http://habrahabr.ru/post/150041> (дата обращения: 13.11.2020)