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 по дисциплине

Объектно-ориентированное программирование

«Организация работы с базой данных»

Наименование темы

Выполнил студент группы ИСТб 19–2 Первых А.А.

шифр Фамилия И.О.

Проверила Маланова Т.В.

Фамилия И.О.

Содержание отчета на 29 стр.

Иркутск 2020 г.

Оглавление

[1 Постановка задачи 3](#_Toc58184130)

[2 Структура меню 3](#_Toc58184131)

[3. Проектирование классов. 4](#_Toc58184132)

[4 Таблица спецификаций 5](#_Toc58184133)

[4.1 Описание класса Staff 5](#_Toc58184134)

[4.2 Описание класса Group 5](#_Toc58184135)

[4.3 Описание класса Administration. 6](#_Toc58184136)

[4.4 Описание класса Engineer. 6](#_Toc58184137)

[4.5 Описание класса Worker. 7](#_Toc58184138)

[5 Разработка тестов 8](#_Toc58184139)

[6 Результаты тестирования 9](#_Toc58184140)

[7. Листинг исходного кода 11](#_Toc58184141)

[Заключение 17](#_Toc58184142)

[Список литературы 18](#_Toc58184143)

1 Постановка задачи

**Цель работы**: Применение объектно-ориентированного подхода к

организации доступа к базе данных (БД).

**Задачи**:

Получение общего представления об объектном подходе к организации

доступа к БД, выполнению запросов и обработке результатов.

Получение практического опыта организации работы с БД средствами

платформы java.

**Содержание задания**

Для выполнения задания необходимо разработать java-приложение с графическим пользовательским интерфейсом позволяющее просматривать и редактировать информацию из базы данных. Работу предваряет организация и наполнение базы данных (БД) в соответствии с предметной областью из индивидуального задания. Графический пользовательский интерфейс должен предоставлять возможности работы с записями основных таблиц БД. Состав используемых компонентов определяется самостоятельно, но для их размещения 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.

2 Описание БД

Duty

Duty ID

Duty

title

rate

Staff

Staff

Staff ID

ID

ID

name

age

Profession ID

Duty ID

Profession

Profession ID

ID

Title

Рисунок 1 – Диаграмма классов БД

Таблица 1 – Описание таблиц БД

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя таблицы | Имя атрибута | Тип | Размер | Описание |
| Duty | Duty ID | INTEGER | 8 байт | Идентификатор обязанности  Ключевое поле |
| title | TEXT | 256 байт | Название обязанности |
| rate | TEXT | 256 байт | Ставка |
| Profession | Profession ID | INTEGER | 8 байт | Идентификатор профессии  Ключевое поле |
| title | TEXT | 256 байт | Название профессии |
| Staff | Staff ID | INTEGER | 8 байт | Идентификатор сотрудника  Ключевое поле |
| name | TEXT | 256 байт | Имя |
| age | TEXT | 256 байт | Возраст |
| Profession ID | INTEGER | 8 байт | Идентификатор профессии |
| Duty ID | INTEGER | 8 байт | Идентификатор обязанности  Ключевое поле |

SQL-запросы к БД

INSERT INTO Profession(title) VALUES(`Администратор`)

INSERT INTO Duty(title,rate) VALUES(`Управление персоналом`,`230`)

INSERT INTO Staff(name,age,Profession\_id,Duty\_id) VALUES(`Дмитрий`,`56`,`1`,`1`)

3 Проектирование графического интерфейса

На главном окне с помощью компоновщика BorderLayout должны быть размещены левая панель (West), центральная панель (Center). На левой панель будут расположены кнопки и их подписи, для этого используем GridLayout.На центральной панели будет расположена таблица.

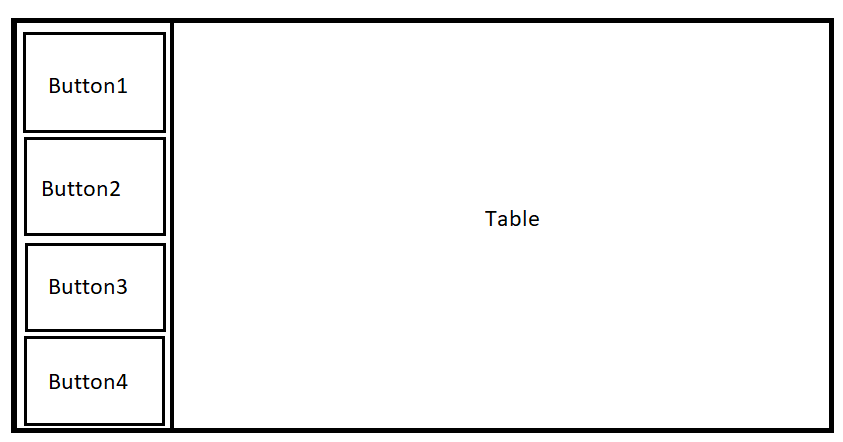


Рисунок 2 – Главное окно программы

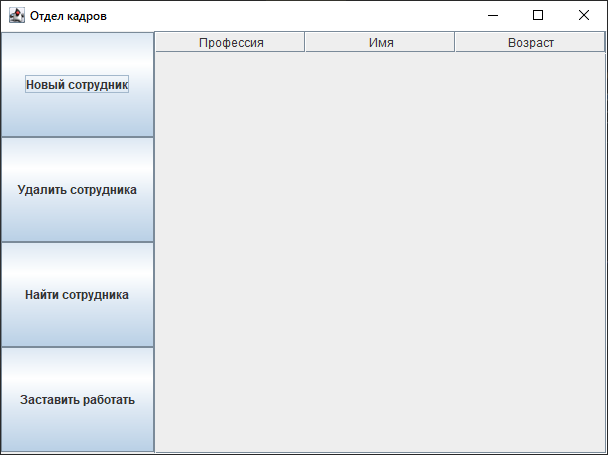
Главное окно программы содержит панель с кнопками и таблицу, отображающую список сотрудников. 

Рисунок 3 – Главное окно программы

Диалоговое окно добавления нового сотрудника, профессия выбирается радиокнопками, а имя и возраст вводится и проверяется вручную.

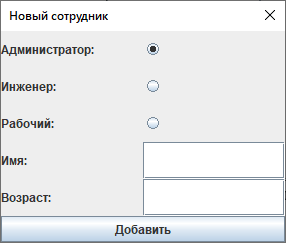


Рисунок 4 –Окно добавления нового сотрудника

Диалоговое окно удаления сотрудника по имени

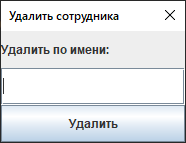


Рисунок 5 –Окно удаления сотрудника

Диалоговое окно поиска сотрудника, при выборе радиокнопки и нажатии на кнопку «Поиск по должности» открывается таблица с результатом поиска, при вводе имени и нажатии на «Поиск по имени», также открывается таблица с результатом.

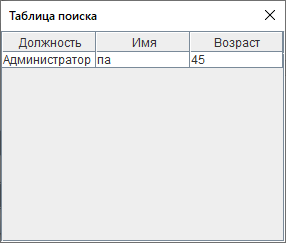
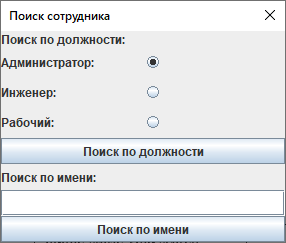


Рисунок 6 – Окно поиска сотрудника и таблица с результатом

Диалоговые сообщения с выводом ошибки при добавлении сотрудника

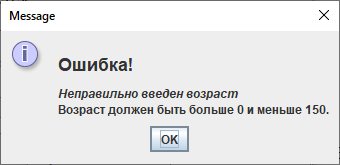
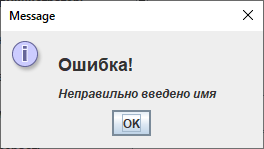


Рисунок 7 – Сообщения обработки корректности ввода добавлении сотрудника

Диалоговые сообщения с выводом ошибки при удалении и поиска сотрудника.

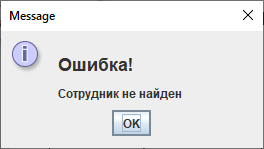


Рисунок 8 – Сообщения обработки исключений удалении и поиска сотрудника

Диалоговые сообщения с выводом ошибки при нажатии на кнопку «Заставить работать» с не выбранной строкой

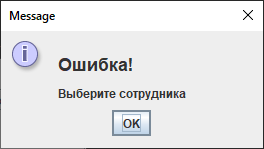


Рисунок 9 – Сообщения обработки исключений при работе сотрудника

3. Проектирование классов.

На рисунке 1 представлена UML диаграмма классов.

Пакет People

Staff

Group

public static ArrayList staff;

public static ArrayList staffSearch;

public static void add;

public static void delete;

public static searchName;

public static searchProff;

public static void getSize;

public static Staff getStaff;

private String name;

private int age;

private String profession;

public Staff;

public abstract void work;

public void getInfo;

public String getName;

public void setName;

public int getAge;

public void setAge;

public String getProfession;

public void setProfession

Engineer

public Engineer;

public void work;

Worker

public Worker;

public void work;

Administration

public Administration;

public void work;

Класс Staff – абстрактный класс кадры служит для выделения общих данных и поведения для других классов.

Класс Group - группирует объекты.

Класс Administration – Класс администратора.

Класс Engineer – Класс инженера.

Класс Worker – Класс рабочего.

Group

Menu\_GUI

private MyTableModel tableModel;

public Menu\_GUI

private class AddButtonEventListener

private class DeleteButtonEventListener

private class SearchButtonEventListener

private class SearchPopupButtonEventListener

public static void infoBox

private void updateTableSearch

Пакет Menu

Menu\_control

Public static staffName;

Public static staffAge

MyTableModel

private Group data

public MyTableModel

public int getRowCount

public int getColumnCount

public String getColumnName

public Object getValueAt

4 Таблица спецификаций

4.1 Описание класса Staff

Таблица 1 – Поля абстрактного класса Staff

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | name | Имя кадра | String | private |
| 2 | age | Возраст кадра | int | private |
| 3 | profession | Профессия кадра | String | private |
| 4 | Duty | Обязанность кадра | String | private |

Таблица 2 – Конструкторы абстрактного класса Staff

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип доступа | Параметры |
| 1 | Staff | Конструктор задания полей. | public | name |
| age |
| profession |

Таблица 3 – Методы класса

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа | Абстрактный/ неабстрактный | Параметры |
| 1 | work | Метод работы | String | public | Абстрактный |  |
| 2 | getInfo | getter информации о кадре | String | public | Абстрактный |  |
| 3 | getName | getter имени | String | public | неабстрактный |  |
| 4 | setName | setter имени | String | public | неабстрактный | name |
| 5 | getAge | getter возраста | int | public | неабстрактный |  |
| 6 | setAge | setter возраста | int | public | неабстрактный | age |
| 7 | getProfession | getter профессии | String | public | неабстрактный |  |
| 8 | setProfession | setter профессии | String | public | неабстрактный | profession |

4.2 Описание класса Group

Таблица 4 – Поля абстрактного класса Group

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа |
| 1 | ArrayList<Staff> staff | Лист объектов | List | Public static |
| 1 | ArrayList<Staff> staffSearch | Лист объектов | List | Public static |

Таблица 5 – Методы класса

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Название | Назначение | Тип данных | Тип доступа | Абстрактный/ неабстрактный | Параметры |
| 1 | add | Добавление объекта | String | public static | неабстрактный | String[] list |
| 2 | delete | Удаление объекта | String | public static | неабстрактный | String name |
| 3 | searchName | Поиск сотрудника по имени | ArrayList<Staff> | public static | неабстрактный | String name |
| 4 | searchProff | Поиск сотрудника по профессии | ArrayList<Staff> | public static | неабстрактный | String proff |
| 5 | getStaff | Получение сотрудника по индексу | Staff | public static | неабстрактный | String name |
| 6 | getSize | Получение размера staff | Staff | public | неабстрактный |  |

5 Разработка тестов

Таблица 15 – Таблица тестов

|  |  |  |
| --- | --- | --- |
| № | Действие | Результат действия |
| 1 | Первоначальный запуск программы и подключение БД | При старте программы появляется главная  Форма «Отдел кадров» |
| 2 | Создание нового кадра | Создания окна (кнопка «Новый сотрудник»), ввод имени, возвраста, и выбор профессии. Добавление сотрудника кнопкой «Добавить». |
| 3 | Удаление кадра | Создания окна (кнопка «Удалить сотрудника»), Удаление кадра в списке по имени. Удаление сотрудника кнопкой «Добавить». |
| 4 | Поиск сотрудника | Создания окна (кнопка «Найти сотрудника»), Поиск кадра в списке по имени. Поиск по должности или имени, кнопкой «Поиск». |
| 5 | Действие кадра | Вывод сообщения о действии кадра при выбранном в таблице сотрудника. Команда «Заставить работать». |
| 6 | Предупреждения при добавлении сотрудника | Сообщения обработки корректности ввода добавлении сотрудника |
| 7 | Предупреждения при отсутсвие кадров | Вывод предупреждения |
| 8 | Отсутстве выбора сотрудника | Вывод предупреждения |

6 Результаты тестирования

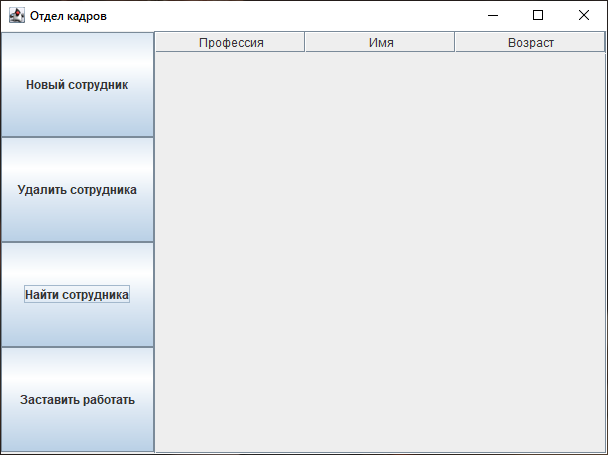


Рисунок 8 – Главное меню программы.

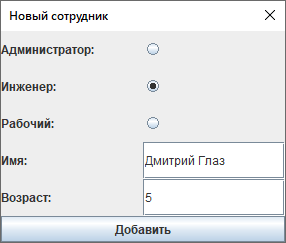


Рисунок 9 – Создание нового кадра.

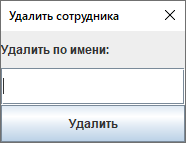


Рисунок 10 – Удаление кадра.

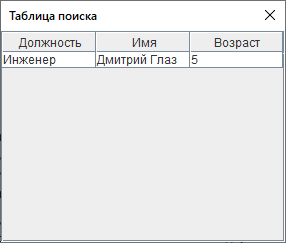
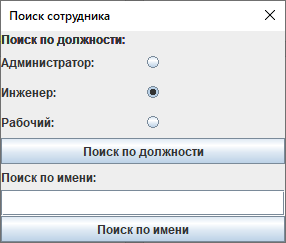


Рисунок 11 – Поиск заданного кадра.

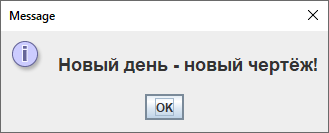


Рисунок 12 – Действие кадра.

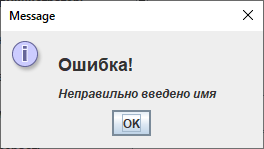
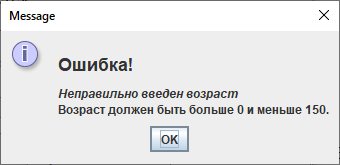
 

Рисунок 13 – Предупреждения при добавлении сотрудника.

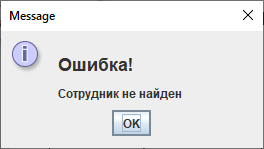


Рисунок 14 – Предупреждения при отсутсвие кадров.

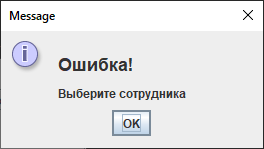


Рисунок 15 – Отсутстве выбора сотрудника.

7. Листинг исходного кода

**Класс Main**

Menu\_GUI GUI = new Menu\_GUI();

GUI.setVisible(true);

**Класс Menu\_control**

public static boolean staffName (String name) {

if (name.matches("^[\\p{L} ]+")) { //Содержание букв и пробел

return true;

}

else {

Menu\_GUI.infoBox("<i>Неправильно введено имя</i>");

}

return false;

}

public static boolean staffAge (String age) {

if (age.matches("-?\\d+(\\d+)?") ) {

if (Integer.parseInt(age) > 0 && Integer.parseInt(age) < 150 ){return true;}

else {Menu\_GUI.infoBox("<i>Неправильно введен возраст</i><p>Возраст должен быть больше 0 и меньше 150.</p>");}

}

else {Menu\_GUI.infoBox("<i>Неправильно введен возраст</i><p>Возраст должен быть больше 0 и меньше 150.</p>");}

return false;

}

**Класс Staff**

private String name;

private int age;

private String profession;

private String work;

public Staff(String name, int age, String profession, String work) {

this.name = name;

this.age = age;

this.profession = profession;

this.work = work;

}

public void getInfo () {

System.out.printf("Имя: %s%n", name);

System.out.printf("Возраст: %d%n", age);

System.out.printf("Профессия: %s%n", profession);

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public String getProfession() {

return profession;

}

public void setProfession(String profession) {

this.profession = profession;

}

**Класс Group**

public static ArrayList<Staff> staff = new ArrayList<Staff>();

public static ArrayList<Staff> staffSearch = new ArrayList<Staff>();

// Метод добавления сотрудника

public static void add (String[] list) throws SQLException {

DBWorker.addStudent(new Staff(list[1],Integer.parseInt(list[2]),list[0],list[3]));

}

// Метод удаление сотрудника

public static void delete(String name) throws SQLException {

int i = -1;

if ((staff.removeIf(nextStaff -> nextStaff.getName().equals(name))) && DBWorker.deleteStudent(name)){

i++;

}

if (i == -1) {

Menu\_GUI.infoBox("Сотрудник не найден");

}

}

// Метод поиска сотрудника

public static ArrayList<Staff> searchName (String name) {

int check = 0;

staffSearch.clear();

for (Staff stf : staff) {

if (name.equals(stf.getName())) {

staffSearch.add(stf);

check++;

}

}

if (check == 0) {Menu\_GUI.infoBox("Сотрудник не найден"); return null;}

else {

return staffSearch;

}

}

public static ArrayList<Staff> searchProff (String proff) {

int check = 0;

staffSearch.clear();

for (Staff stf : staff) {

if (proff.equals(stf.getProfession())) {

staffSearch.add(stf);

check++;

}

}

if (check == 0) {Menu\_GUI.infoBox("Сотрудник не найден"); return null;}

else {

return staffSearch;

}

}

public static String[] getWorkStaff(Staff staff) throws SQLException {

int workId = DBWorker.getWorkID(staff);

return DBWorker.getDuty(workId);

}

public int getSize(){

return staff.size();

}

public static Staff getStaff(int index){

return staff.get(index);

}

**Класс DBWorker**

public static final String PATH\_TO\_DB\_FILE = "database.db";

public static final String URL = "jdbc:sqlite:" + PATH\_TO\_DB\_FILE;

public static Connection conn;

public static void initDB() {

try {

conn = DriverManager.getConnection(URL);

if (conn != null) {

DatabaseMetaData meta = conn.getMetaData();

System.out.println("Драйвер: " + meta.getDriverName());

//createDB();

DBWorker.getAllStudents();

}

} catch (SQLException ex) {

System.out.println("Ошибка подключения к БД: " + ex);

}

}

public static void closeConnection() throws SQLException {

if (conn != null) {

conn.close();

}

}

public static void createDB() throws SQLException {

Statement statmt = conn.createStatement();

statmt.execute("CREATE TABLE if not exists 'Profession' ('id' INTEGER PRIMARY KEY AUTOINCREMENT, 'title' text);");

System.out.println("Таблица создана или уже существует.");

statmt.execute("CREATE TABLE if not exists 'Duty' ('id' INTEGER PRIMARY KEY AUTOINCREMENT, 'title' text, 'rate' text);");

System.out.println("Таблица создана или уже существует.");

statmt.execute("CREATE TABLE if not exists 'Staff' ('id' INTEGER PRIMARY KEY AUTOINCREMENT, 'name' text, 'age' INTEGER, 'Profession\_id' INTEGER NOT NULL, 'Duty\_id' INTEGER NOT NULL, FOREIGN KEY (Profession\_id) REFERENCES Profession (id), FOREIGN KEY (Duty\_id) REFERENCES Duty (id));");

System.out.println("Таблица создана или уже существует.");

}

public static void addProfession(String title) throws SQLException {

PreparedStatement statement = conn.prepareStatement(

"INSERT INTO Profession(`title`) " +

"VALUES(?)");

statement.setString(1, title);

statement.execute();

statement.close();

}

public static ArrayList<String> getAllProfession() throws SQLException {

ArrayList<String> allProfession = new ArrayList<String>();

Statement statement = conn.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT id, title FROM Profession");

while (resultSet.next()) {

allProfession.add(resultSet.getString("title"));

System.out.println(resultSet.getInt("id") + " " + resultSet.getString("title"));

}

resultSet.close();

statement.close();

return allProfession;

}

public static int getProfessionId(String profTitle) throws SQLException {

Statement statement = conn.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT id FROM Profession WHERE Profession.title ='"+profTitle+"'");

int profId = -1;

profId = resultSet.getInt(1);

resultSet.close();

statement.close();

return profId;

}

public static String getProfessionName(int profId) throws SQLException {

String profTitle = "";

Statement statement = conn.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT title FROM Profession WHERE Profession.id ='"+profId+"'");

profTitle = resultSet.getString(1);

resultSet.close();

statement.close();

return profTitle;

}

public static void addDuty(String title, String rate) throws SQLException {

PreparedStatement statement = conn.prepareStatement(

"INSERT INTO Duty(title,rate) " +

"VALUES(?,?)");

statement.setString(1, title);

statement.setString(2, rate);

statement.execute();

statement.close();

}

public static int getDutyId(String dutyTitle) throws SQLException {

Statement statement = conn.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT id FROM Duty WHERE Duty.title ='"+dutyTitle+"'");

int dutyId = -1;

dutyId = resultSet.getInt(1);

resultSet.close();

statement.close();

return dutyId;

}

public static String getDutyName(int dutyId) throws SQLException {

String dutyTitle = "";

Statement statement = conn.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT title FROM Duty WHERE Duty.id ='"+dutyId+"'");

dutyTitle = resultSet.getString(1);

resultSet.close();

statement.close();

return dutyTitle;

}

public static String[] getDuty(int dutyId) throws SQLException {

String[] dutyTitle = new String[2];

Statement statement = conn.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT title, rate FROM Duty WHERE Duty.id ='"+dutyId+"'");

dutyTitle[0] = resultSet.getString(1);

dutyTitle[1] = resultSet.getString(2);

resultSet.close();

statement.close();

return dutyTitle;

}

public static int getWorkID(Staff staff) throws SQLException {

Statement statement = conn.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT Duty\_id FROM Staff WHERE Staff.name ='"+staff.getName()+"' AND Staff.age ='"+staff.getAge()+"'");

int workId = resultSet.getInt(1);

resultSet.close();

statement.close();

return workId;

}

public static ArrayList<String> getAllDuty() throws SQLException {

ArrayList<String> allWork = new ArrayList<String>();

Statement statement = conn.createStatement();

ResultSet resultSet = statement.executeQuery("SELECT id, title, rate FROM Duty");

while (resultSet.next()) {

allWork.add(resultSet.getString("title"));

}

resultSet.close();

statement.close();

return allWork;

}

public static void addStudent(Staff staff) throws SQLException {

PreparedStatement statement = conn.prepareStatement(

"INSERT INTO Staff(name,age,Profession\_id,Duty\_id) " +

"VALUES(?,?,?,?)");

statement.setObject(1, staff.getName());

statement.setObject(2, staff.getAge());

statement.setObject(3, getProfessionId(staff.getProfession()));

statement.setObject(4, getDutyId(staff.getWork()));

statement.execute();

statement.close();

}

public static void getAllStudents() throws SQLException {

try (Statement statement = conn.createStatement()) {

ResultSet resultSet = statement.executeQuery("SELECT Staff.name, Staff.age, Staff.Profession\_id, Staff.Duty\_id, Profession.title, Duty.title, Duty.rate FROM Staff JOIN Profession,Duty ON Profession.id = Staff.Profession\_id AND Duty.id = Staff.Duty\_id");

Group.staff.clear();

while (resultSet.next()) {

Group.staff.add(new Staff(resultSet.getString("name"), resultSet.getInt("age"), getProfessionName(resultSet.getInt("Profession\_id")), getDutyName(resultSet.getInt("Duty\_id")) ));

}

resultSet.close();

} catch (SQLException e) {

e.printStackTrace();

// Если произошла ошибка - возвращаем пустую коллекцию

}

}

public static boolean deleteStudent(String name) throws SQLException {

Statement statement = conn.createStatement();

statement.execute("DELETE FROM Staff WHERE Staff.name ='"+name+"'");

statement.close();

return true;

}

**Класс Menu\_GUI**

private MyTableModel tableModel;

private DefaultTableModel tableModelSearch;

private JTable table1;

private JTable tableSearch;

private JDialog popupNew = new JDialog(this, "Новый сотрудник", true);

private JDialog popupSearch = new JDialog(this, "Поиск сотрудника", true);

private JDialog popupDelete = new JDialog(this, "Удалить сотрудника", true);

private JDialog popupTableSearch = new JDialog(this, "Таблица поиска", true);

private JPanel buttonsPanel = new JPanel();

private JButton newStaff = new JButton("Новый сотрудник");

private JButton deleteStaff = new JButton("Удалить сотрудника");

private JButton searchStaff = new JButton("Найти сотрудника");

private JButton workStaff = new JButton("Заставить работать");

private Object[] columnsHeader = new String[] {"Должность", "Имя",

"Возраст"};

public Menu\_GUI() {

super("Отдел кадров");

this.setBounds(300, 500, 550, 600);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

DBWorker.initDB();

this.addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

try {

DBWorker.closeConnection();

} catch (SQLException ex) {

ex.printStackTrace();

}

}

});

tableModel = new MyTableModel(new Group());

table1 = new JTable(tableModel);

// Создание кнопки добавления сотрудника

newStaff.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

AddButtonEventListener addButtonEventListener=new AddButtonEventListener();

} catch (SQLException throwables) {

throwables.printStackTrace();

}

}

});

// Создание кнопки удаления сотрудника

deleteStaff.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

DeleteButtonEventListener DeleteButtonEventListener=new DeleteButtonEventListener();

}

});

// Поиск сотрудника

searchStaff.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try {

SearchButtonEventListener SearchButtonEventListener=new SearchButtonEventListener();

} catch (SQLException throwables) {

throwables.printStackTrace();

}

}

});

// Заставить сотрудника работать

workStaff.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try{

String staffWork[] = Group.getWorkStaff(Group.getStaff(table1.getSelectedRow()));

JOptionPane.showMessageDialog(null, "<html><h2>Обязанность: "+staffWork[0]+"</h2><h3>Ставка:"+staffWork[1]+"</h3>");

}

catch (IndexOutOfBoundsException | SQLException ex){

infoBox("Выберите сотрудника");

}

}

});

// Формирование интерфейса

buttonsPanel.setLayout(new GridLayout(4,1));

buttonsPanel.add(newStaff);

buttonsPanel.add(deleteStaff);

buttonsPanel.add(searchStaff);

buttonsPanel.add(workStaff);

getContentPane().add(buttonsPanel, BorderLayout.WEST);

Box contents = new Box(BoxLayout.Y\_AXIS);

contents.add(new JScrollPane(table1));

getContentPane().add(contents);

this.pack();

}

// Создание кнопки добавления работника

private class AddButtonEventListener extends JDialog {

private JPanel staffPanel = new JPanel();

private JPanel addPanel = new JPanel();

private JPanel profPanel = new JPanel();

private JPanel workPanel = new JPanel();

private JLabel labelName = new JLabel("Имя:");

private JLabel labelWork = new JLabel("Обязанность:");

private JLabel labelProf = new JLabel("Профессия:");

private JLabel labelAge = new JLabel("Возраст:");

private JTextField textName = new JTextField(5);

private JTextField textAge = new JTextField(5);

private String profSelect, workSelect, name, age;

public AddButtonEventListener() throws SQLException {

JButton subButton = new JButton("Добавить");

popupNew.setPreferredSize(new Dimension(400,350));

//Панель выбора профессии

ButtonGroup groupProf = new ButtonGroup();

profPanel.setLayout(new GridLayout(2,1));

for (String data : DBWorker.getAllProfession()){

System.out.println(data);

final JRadioButton button1 = new JRadioButton(data);

button1.setActionCommand(data);

groupProf.add(button1);

profPanel.add(button1);

}

//Панель выбора обязанности

ButtonGroup groupWork = new ButtonGroup();

workPanel.setLayout(new GridLayout(2,1));

for (String data : DBWorker.getAllDuty()){

System.out.println(data);

final JRadioButton button1 = new JRadioButton(data);

button1.setActionCommand(data);

groupWork.add(button1);

workPanel.add(button1);

}

addPanel.setLayout(new GridLayout(4,1));

addPanel.add(labelProf);

addPanel.add(profPanel);

addPanel.add(labelWork);

addPanel.add(workPanel);

staffPanel.setLayout(new GridLayout(2,2));

staffPanel.add(labelName);

staffPanel.add(textName);

staffPanel.add(labelAge);

staffPanel.add(textAge);

popupNew.add(new JScrollPane(addPanel), BorderLayout.NORTH);

popupNew.add(staffPanel, BorderLayout.CENTER);

popupNew.add(subButton, BorderLayout.SOUTH);

// Добавление нового сотрудника при нажатии на кнопку

subButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

name = textName.getText();

age = textAge.getText();

//Проверка имени

boolean nameTrue = Menu\_control.staffName(name);

//Проверка возраста

boolean ageTrue = Menu\_control.staffAge(age);

try{

profSelect = groupProf.getSelection().getActionCommand();

workSelect = groupWork.getSelection().getActionCommand();

}

catch (NullPointerException ex){

infoBox("Выберите профессия и/или обязанность");

}

if (nameTrue && ageTrue && (profSelect != null) && (workSelect != null)) {

String[] staff = new String[]{profSelect, name, age, workSelect};

// Добавление нового сотрудника

try {

Group.add(staff);

} catch (SQLException ex) {

ex.printStackTrace();

}

tableModel.fireTableDataChanged();

textName.setText("");

textAge.setText("");

name = "";

age = "";

}

}

}

);

popupNew.setLocationRelativeTo(null);

popupNew.setResizable(false);

popupNew.pack();

popupNew.setVisible(true);

popupNew.setDefaultCloseOperation(WindowConstants.DISPOSE\_ON\_CLOSE);

}

}

// Удаление сотрудника

private class DeleteButtonEventListener extends JDialog {

private JPanel deletePanel = new JPanel();

private JLabel labelDelete = new JLabel("Удалить по имени:");

private JTextField textName = new JTextField(5);

private JButton deleteButton = new JButton("Удалить");

public DeleteButtonEventListener() {

popupDelete.setPreferredSize(new Dimension(200,150));

deletePanel.setLayout(new GridLayout(3,1));

deletePanel.add(labelDelete);

deletePanel.add(textName);

deletePanel.add(deleteButton);

popupDelete.add(deletePanel, BorderLayout.CENTER);

deleteButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

//Удаление сотрудника по имени

try {

Group.delete(textName.getText());

} catch (SQLException throwables) {

throwables.printStackTrace();

}

tableModel.fireTableDataChanged();

textName.setText("");

}

});

popupDelete.setLocationRelativeTo(null);

popupDelete.setResizable(false);

popupDelete.setDefaultCloseOperation(WindowConstants.DISPOSE\_ON\_CLOSE);

popupDelete.pack();

popupDelete.setVisible(true);

}

}

// Создание кнопки поиска работника

private class SearchButtonEventListener extends JDialog {

private JPanel staffPanel = new JPanel();

private JPanel namePanel = new JPanel();

private JLabel labelWork = new JLabel("Поиск по должности:");

private JLabel labelName = new JLabel("Поиск по имени:");

private JTextField textName = new JTextField(5);

private JButton workButton = new JButton("Поиск по должности");

private JButton nameButton = new JButton("Поиск по имени");

private String radioButtonSelect;

public SearchButtonEventListener() throws SQLException {

popupTableSearch.setPreferredSize(new Dimension(300,250));

tableModelSearch = new DefaultTableModel(columnsHeader, 0);

tableModelSearch.setColumnIdentifiers(columnsHeader);

popupSearch.setPreferredSize(new Dimension(300,250));

ButtonGroup groupProf = new ButtonGroup();

staffPanel.setLayout(new GridLayout(2,1));

for (String data : DBWorker.getAllProfession()){

System.out.println(data);

final JRadioButton button1 = new JRadioButton(data);

button1.setActionCommand(data);

groupProf.add(button1);

staffPanel.add(button1);

}

popupSearch.add(labelWork, BorderLayout.NORTH);

popupSearch.add(new JScrollPane(staffPanel), BorderLayout.CENTER);

namePanel.setLayout(new GridLayout(4,1));

namePanel.add(workButton);

namePanel.add(labelName);

namePanel.add(textName);

namePanel.add(nameButton);

popupSearch.add(namePanel, BorderLayout.SOUTH);

SearchPopupButtonEventListener SearchPopupButtonEventListener=new SearchPopupButtonEventListener();

workButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

try{

radioButtonSelect = groupProf.getSelection().getActionCommand();

}

catch (NullPointerException ex){

infoBox("Выберите профессия и/или обязанность");

}

if(Group.searchProff(radioButtonSelect) != null){

updateTableSearch();

}

}

});

nameButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if(Group.searchName(textName.getText()) != null){

updateTableSearch();

}

textName.setText("");

}

});

popupSearch.setLocationRelativeTo(null);

popupSearch.setResizable(false);

popupSearch.pack();

popupSearch.setVisible(true);

popupSearch.setDefaultCloseOperation(WindowConstants.DISPOSE\_ON\_CLOSE);

}

}

private class SearchPopupButtonEventListener extends JDialog {

public SearchPopupButtonEventListener() {

tableSearch = new JTable(tableModelSearch);

popupTableSearch.add(new JScrollPane(tableSearch));

popupTableSearch.setLocationRelativeTo(null);

popupTableSearch.setResizable(false);

popupTableSearch.pack();

popupTableSearch.setDefaultCloseOperation(popupTableSearch.DISPOSE\_ON\_CLOSE);

}

}

// Вывод сообщения

public static void infoBox(String infoMessage) {

JOptionPane.showMessageDialog(null, "<html><h2>Ошибка!</h2>" + infoMessage);

}

// Обновление таблицы поиска

private void updateTableSearch(){

tableModelSearch.setRowCount(0);

for (int i = 0; i < Group.staffSearch.size(); i++){

String staff = Group.staffSearch.get(i).getProfession();

String name = Group.staffSearch.get(i).getName();

int age = Group.staffSearch.get(i).getAge();

tableModelSearch.addRow(new Object[]{staff, name, age});

}

popupTableSearch.setVisible(true);

}

Заключение

В процессе выполнения лабораторной работы 4: “Разработка консольного приложения” были разработаны алгоритмы решения индивидуального задания. Было сформировано 9 тестов, покрывающих все ситуации. Составленные на языке Java программы, реализующие разработанные алгоритмы, прошла все тесты успешно.

В ходе лабораторной работы были получены практические навыки использования конструкций языка Java, а также знакомство с платформой.

Все цели практики достигнуты.

Список литературы

1. Java. Экспресс-курс [электронный ресурс] // Сайт Александра Климова [сайт], URL: http://developer.alexanderklimov.ru/android/java/java.php

2. API Specification for the Java 7 SE. [официальный сайт] URL: http://docs.oracle.com/javase/7/docs/api/

3. The Java Tutorials. SE [электронный ресурс], URL: http://docs.oracle.com/javase/tutorial/index.html

4. Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированны анализ и проек-тирование с примерами приложений. Третье издание. М.: "Вильямс", 2010.

5. Хабибуллин И.Ш. Java 7: для программистов / И. Ш. Хабибуллин. – Санкт-Петербург : БХВ–Петербург, 2014.