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

Центр программной инженерии

ОТЧЕТ

к лабораторной работе №5 по дисциплине:

|  |
| --- |
| «Объектно-ориентированное программирование» |
| Разработка приложения с графическим  Пользовательским интерфейсом |

наименование темы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы: | ИСТб-19-2 |  |  |  | Карбушев И.В. |
|  | шифр группы |  | подпись |  | Фамилия ИО |
| Проверил: | доцент |  |  |  | Маланова Т.В. |
|  | должность |  | подпись |  | Фамилия ИО |

Иркутск. 2020 г.

Оглавление

[Оглавление 2](#_Toc67769318)

[1 Постановка задачи 3](#_Toc67769319)

[2 Проектирование графического интерфейса 4](#_Toc67769320)

[3 Проектирование классов 9](#_Toc67769321)

[4 Проектирование тестов 11](#_Toc67769322)

[5 Листинг программы на Java 23](#_Toc67769323)

[6 Список литературы 45](#_Toc67769324)

1 Постановка задачи

**Вариант - 7**

**Цель работы:** Применение объектно-ориентированного подхода в разработке приложений с графическим интерфейсом.

**Задачи:**

- Получение общего представления о шаблоне программирования Модель/Вид/Контроллер и его практического применения;

- Получение практического опыта работы с базовыми графическими библиотеками java.

**Содержание задания**

Для выполнения задания необходимо разработать графический пользовательский интерфейс средствами java для работы с классами реализованными при выполнении либо 3-ей, либо 4-ой лабораторной работы.

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. Для этого может понадобиться модифицировать программный код из предыдущих  
лабораторных работ.

2 Проектирование графического интерфейса

Большую часть экрана занимает таблица, отображающая данные студентов. Справа отображены кнопки для добавления, удаления, изменения данных с таблицы, для отображения дополнительных данных (оценок учащихся), для поиска студентов по установленной маске, чтения данных из файла и запись их туда.

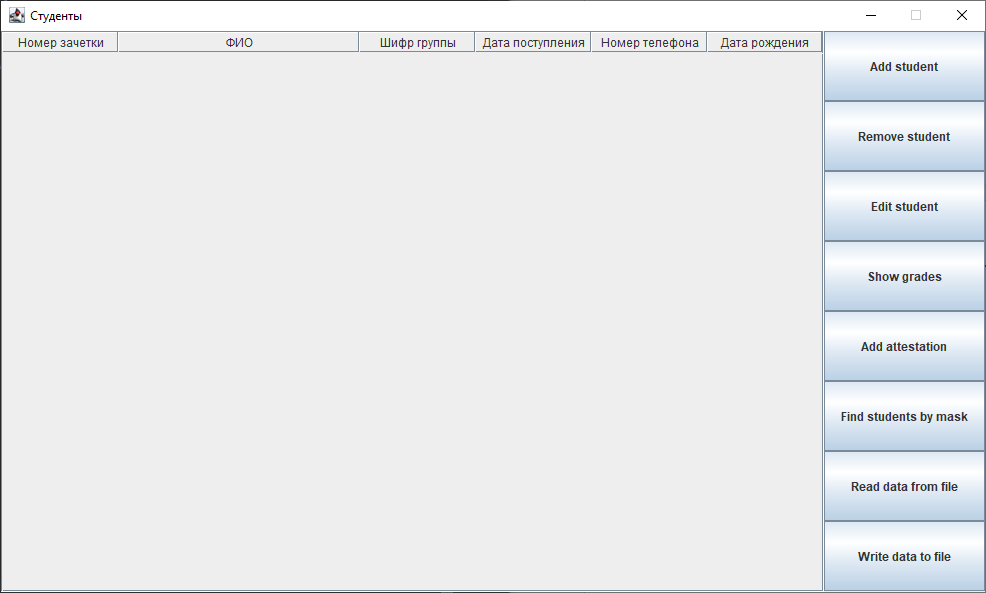


Рисунок 1 – Главное окно программы

В окне представлены поля, введя информацию в которых и нажав enter или кнопку «Принять», в таблицу добавится запись о студенте. Можно также отменить добавление записи с помощью кнопки «Отмена».

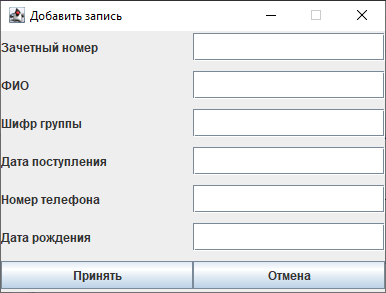


Рисунок 2 – Окно добавления записи о студенте

Если запись не выделена и будет нажата одна из кнопок: «Remove student», «Edit student», «Add attestation», «Show grades», то появится следующее диалоговое окно.

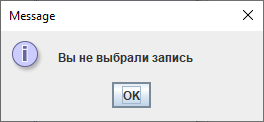


Рисунок 3 - Окно ошибки, когда запись не выбрана

При редактировании выделенной записи в полях будут написаны данные этой самой записи, которые можно изменить и применить.

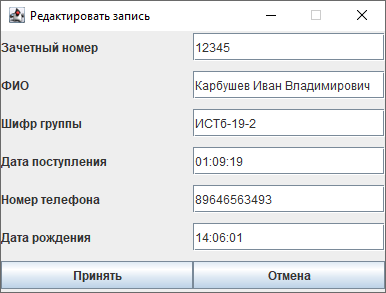


Рисунок 4 – Редактирование записи

При нажатии на кнопку «Show grades», появится таблица с данными оценок выделенного студента. Только сперва нужно добавить записи об оценках.

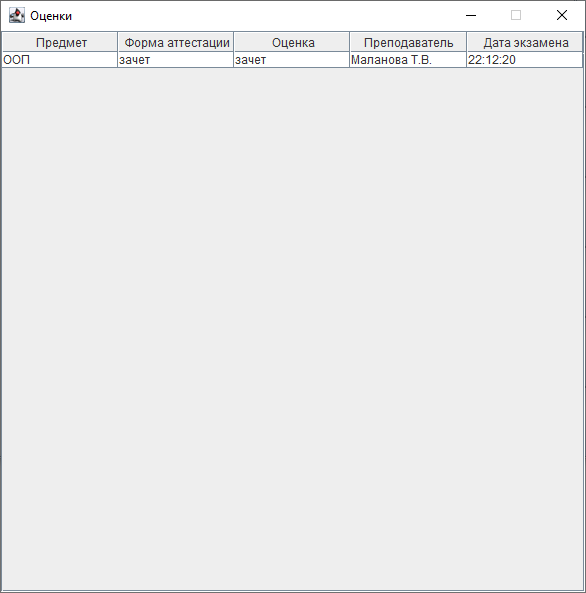


Рисунок 5 – Показ оценок студентов

Введя информацию в поля, выделенному студенту добавится аттестация.

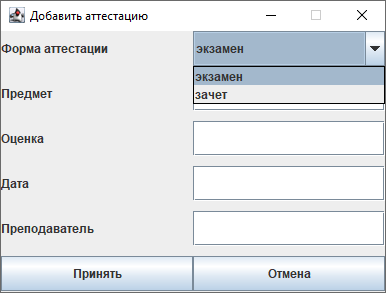


Рисунок 6 – Добавление аттестации

В поля можно ввести маску, по которой найдутся подходящие студенты.

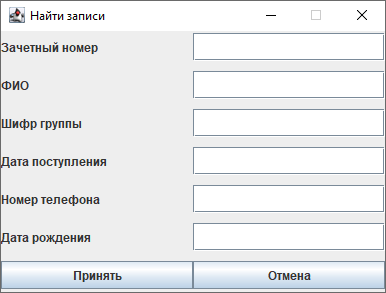


Рисунок 7 – Поиск студентов по маске

Можно добавить данные в таблицу из файла.

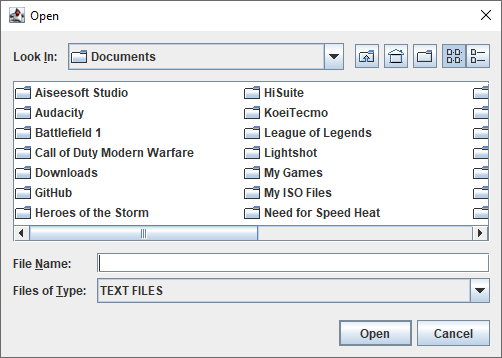


Рисунок 8 – Открытие файла

Также можно сохранить данные из таблицы в файл.

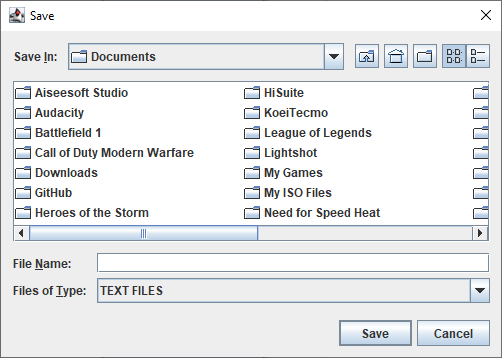
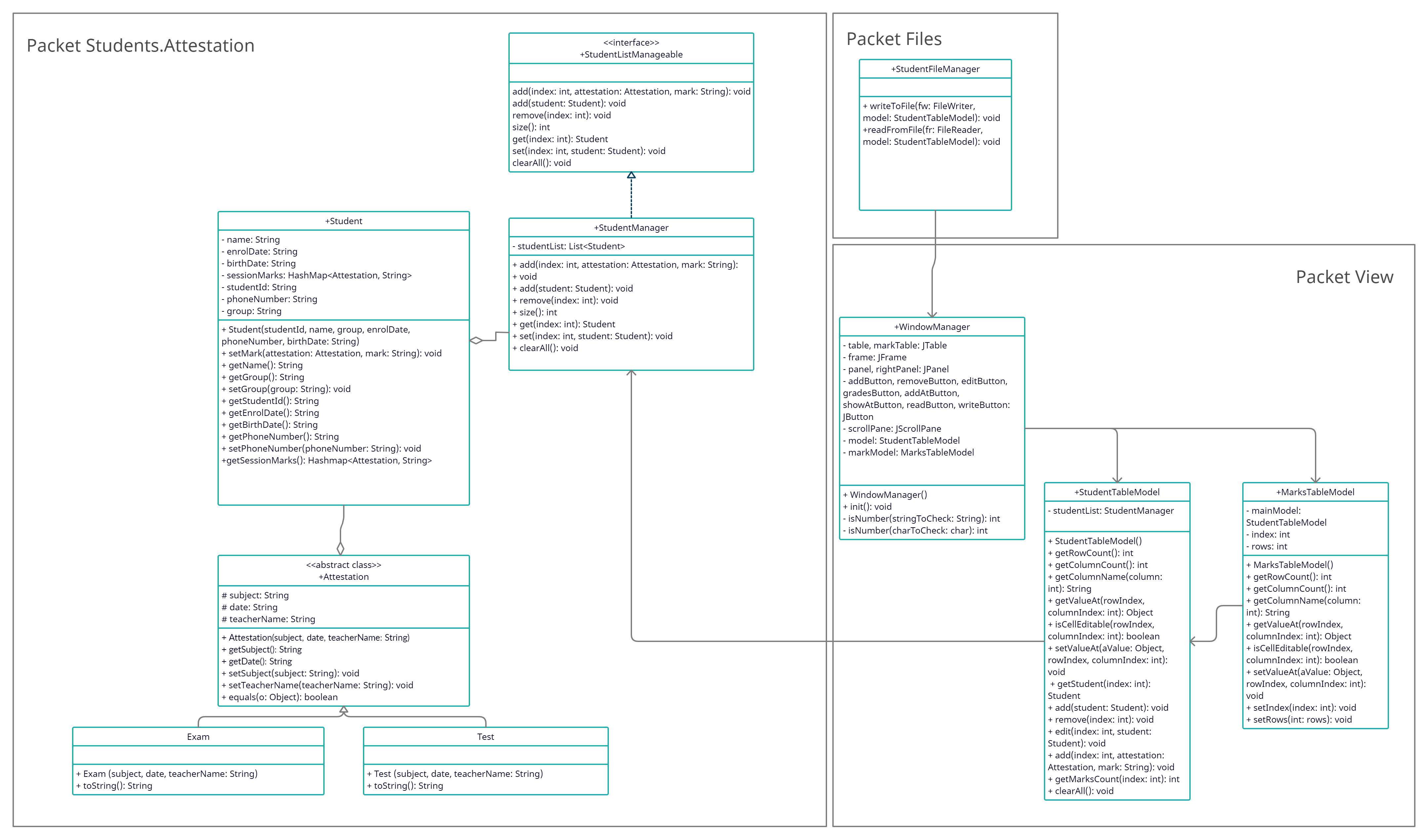


Рисунок 9 – Сохранение в файл

3 Проектирование классов

Рисунок 10 – UML-диаграмма классов



Классы Exam и Test – дочерние классы Attestation

Класс Student – класс-сущность, представляющая студента, агрегирует классом Attestation

Класс StudentManager – класс-группа для Student, так же реализует интерфейс StudentListManageable

Класс WindowManager содержит в себе экземпляры моделей таблиц StudentTableModel и MarksTableModel

Класс StudentFileManager содержит экземпляр класса WindowManager

Класс MarksModelTable содержит экземпляр класса StudentTableModel

Класс StudentTableModel содержит экземпляр класса StudentManager

Рисунок 11 – UML-диаграмма внутренних классов класса WindowManager



Все внутренние классы представляют собой слушателей кнопок класса WindowManager, кроме класса DateKeyAdapter – он представляет собой контроль ввода пользователя в поля ввода текста.

4 Проектирование тестов

|  |  |  |
| --- | --- | --- |
| № | Действие | Результат действия |
| 1 | Первоначальный запуск программы | При старте программы появляется главная форма приложения (Рис. 12) |
| 2 | Добавление новой записи о студенте в таблицу | Запись о студенте добавлена в таблицу (Рис. 13, 14) |
| 3 | Добавление студента с именем, содержащем числа | Ошибка – нельзя вводить числа в имена (Рис. 15) |
| 4 | Удаление студента из таблицы | Запись о студенте успешно удалена (Рис. 16, 17) |
| 5 | Удаление студента из таблицы, когда запись не выделена | Ошибка – запись не выделена (Рис. 18) |
| 6 | Редактирование записи о студенте | Запись успешно отредактирована (Рис. 19, 20) |
| 7 | Добавление записи о аттестации студента | Запись успешно добавлена (Рис. 21) |
| 8 | Добавление записи о аттестации студента, когда запись о студенте не выделена | Ошибка – запись не выделена (Рис. 22) |
| 9 | Показ аттестаций студента | Показ успешно выполнен (Рис. 23) |
| 10 | Показ аттестаций студента, когда запись о студенте не выделена | Ошибка – запись не выделена (Рис. 24) |
| 11 | Поиск студента по маске | Поиск успешно выполнен (Рис. 25, 26) |
| 12 | Чтение записей о студентах и их аттестаций из файла | Чтение успешно выполнено (Рис. 27, 28, 29) |
| 13 | Чтение записей о студентах и их аттестаций из файла, когда файла больше не существует | Ошибка – файла не существует (Рис. 30) |
| 14 | Чтение записей о студентах и их аттестаций из файла, когда данные файле некорректны | Ошибка – формат данных некорректен (Рис. 31, 32) |
| 15 | Сохранение записей о студентах и их аттестациях в файл | Сохранение успешно выполнено (Рис. 33, 34) |
| 16 | При установке количества месяцев в поле «дата…», если их количество равно 0 или больше 12, то автоматически ставится 1 и 12 соответственно. | Результаты тестов не получится показать на картинках, так как изменение даты происходит моментально после ввода некорректных данных |
| 17 | При установке количества дней в месяце равным 0, их количество станет равно 1. |
| 18 | При установке количества дней в месяце не согласно месяцу, их количество будет автоматически откорректировано.  Например, январь – 31 день, апрель - 30 дней, февраль – 28 |

Результаты тестирования

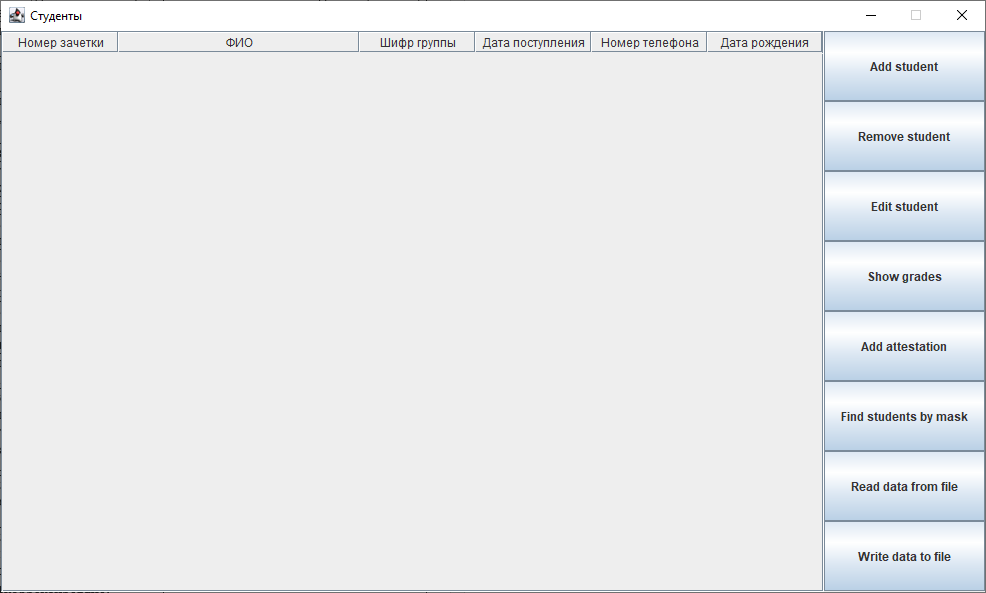


Рисунок 12 – Главная форма приложения

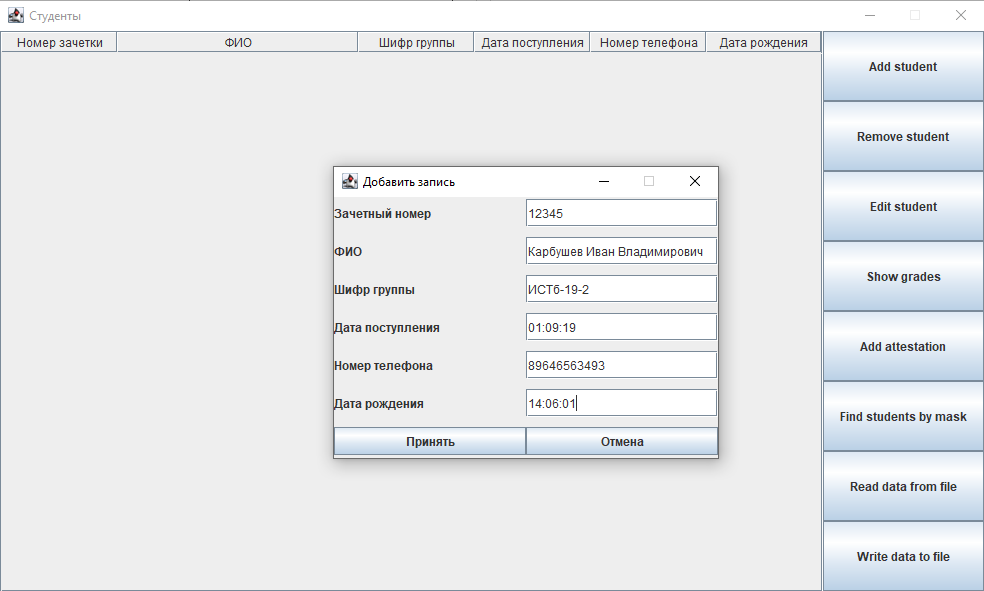


Рисунок 13 – Добавление записи о студенте

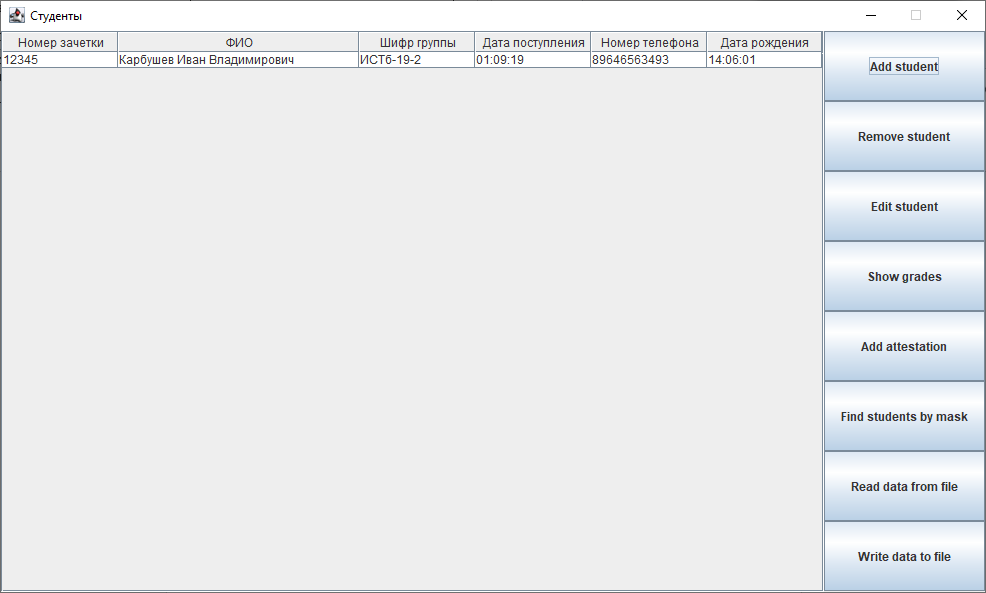


Рисунок 14 – Запись о студенте добавлена

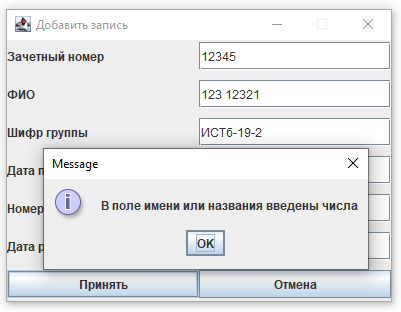


Рисунок 15 – В поле имени введены числа

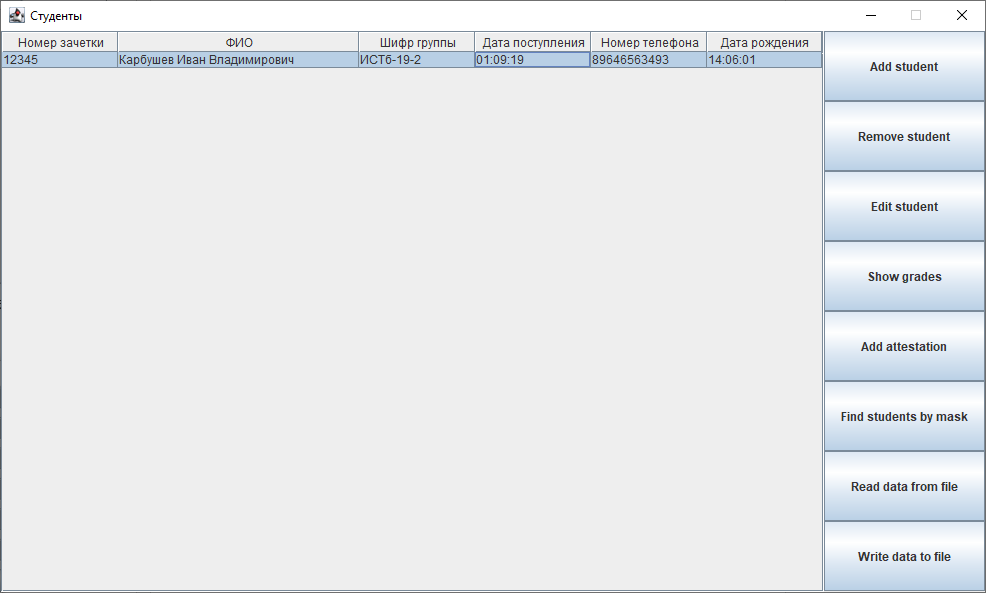


Рисунок 16 – Данные в таблице до удаления записи

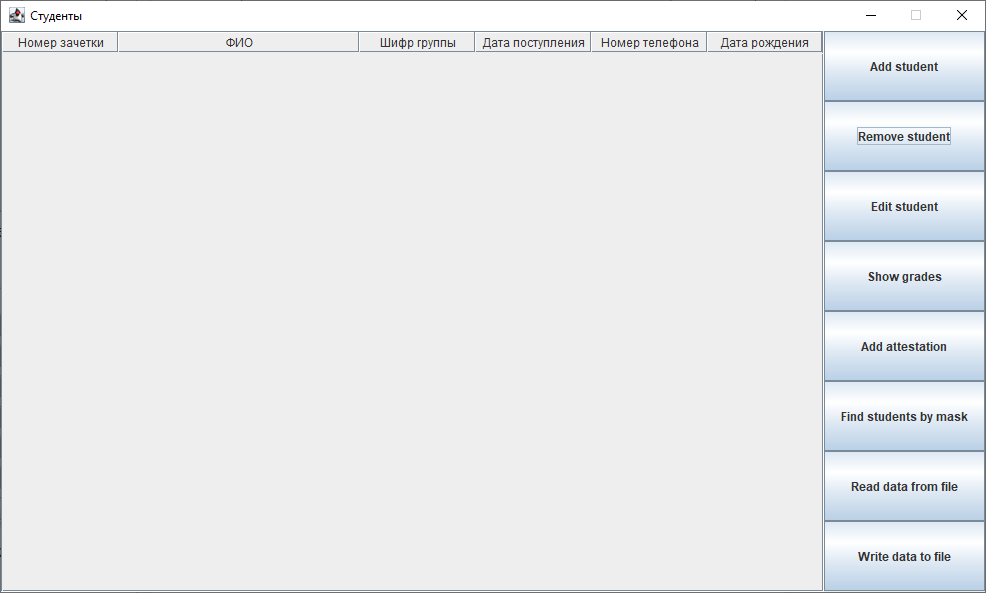


Рисунок 17 – Данные в таблице после удаления записи

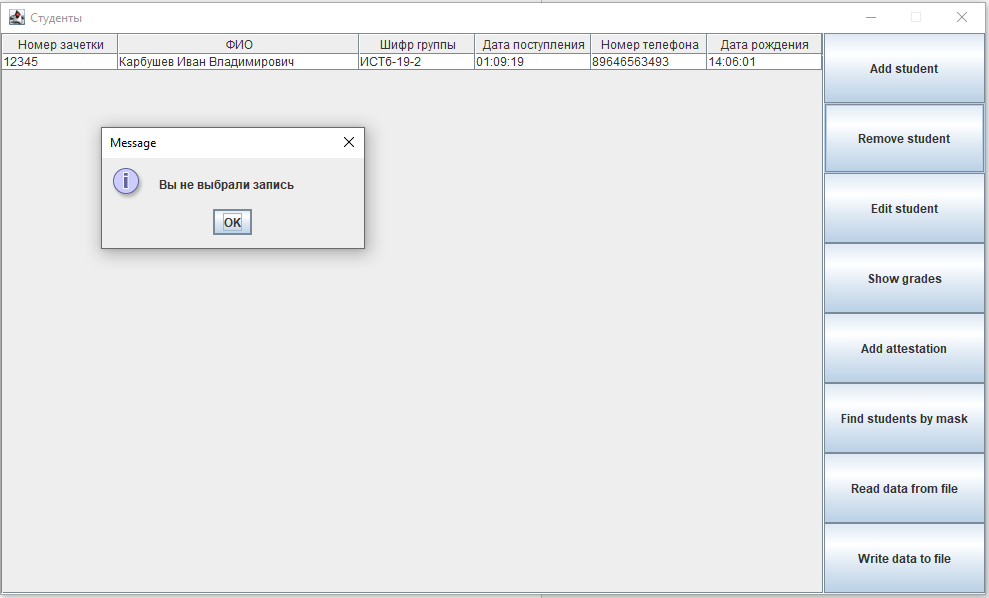


Рисунок 18 – Запись для удаления не выделена

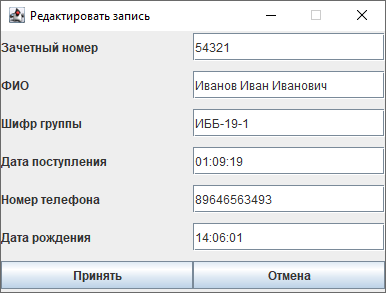


Рисунок 19 – Поле редактирования записи

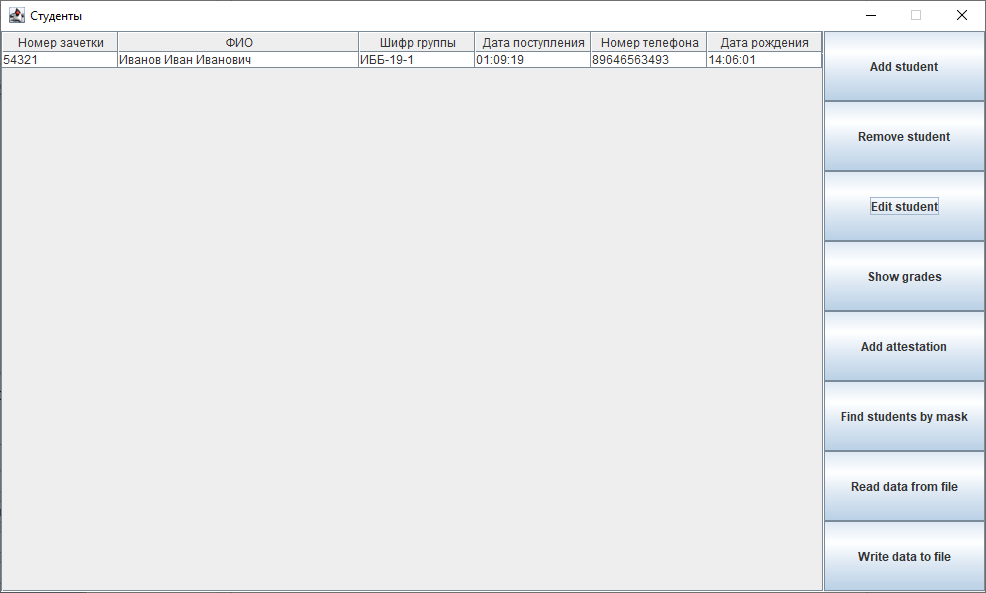


Рисунок 20 – Запись в таблице отредактировано

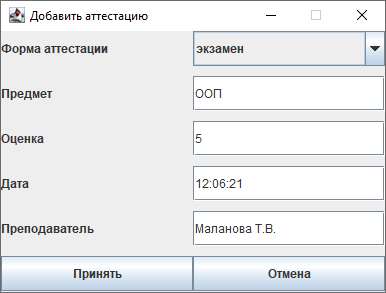


Рисунок 21 – Добавление аттестации студенту

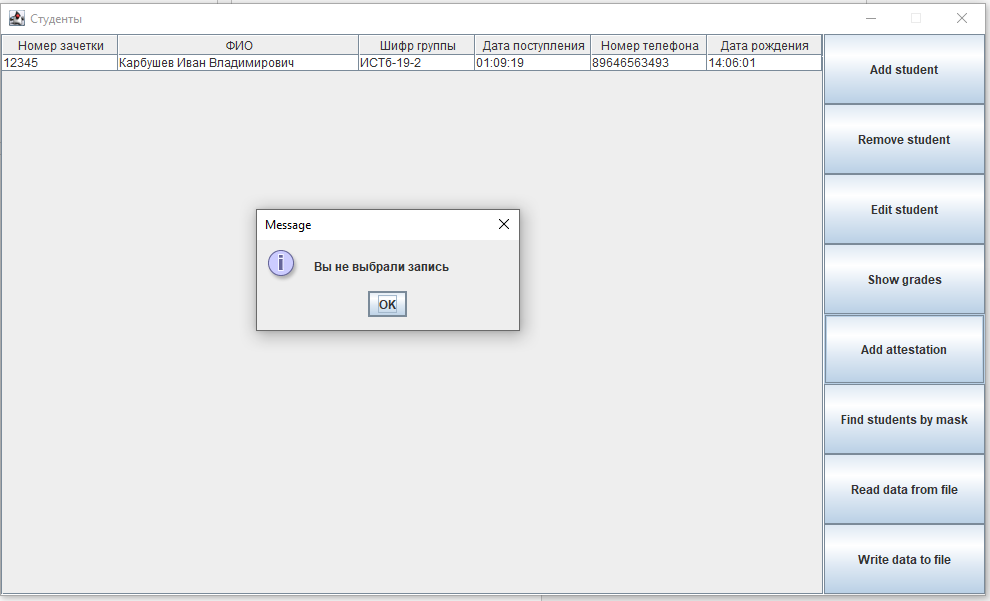


Рисунок 22 – запись не выбрана для добавления аттестации

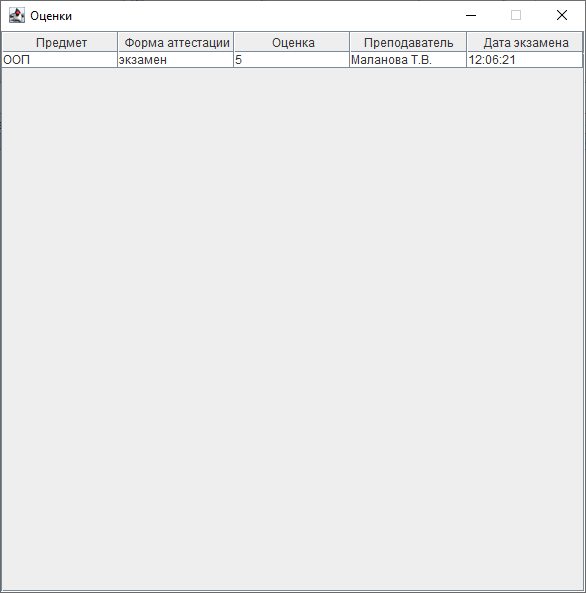


Рисунок 23 – Показ аттестаций выделенного студента

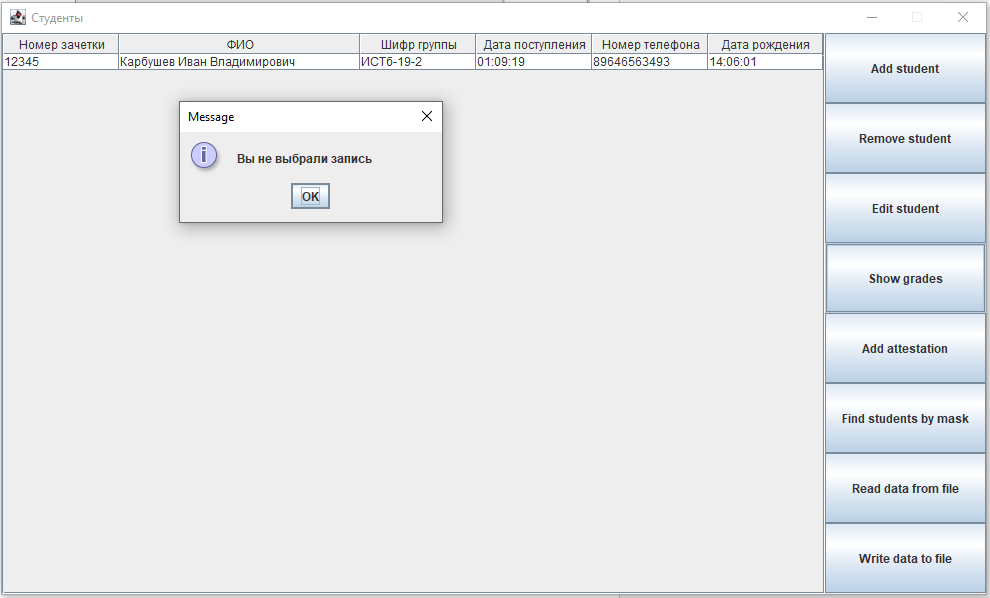


Рисунок 24 – Запись не выделена для показа аттестаций

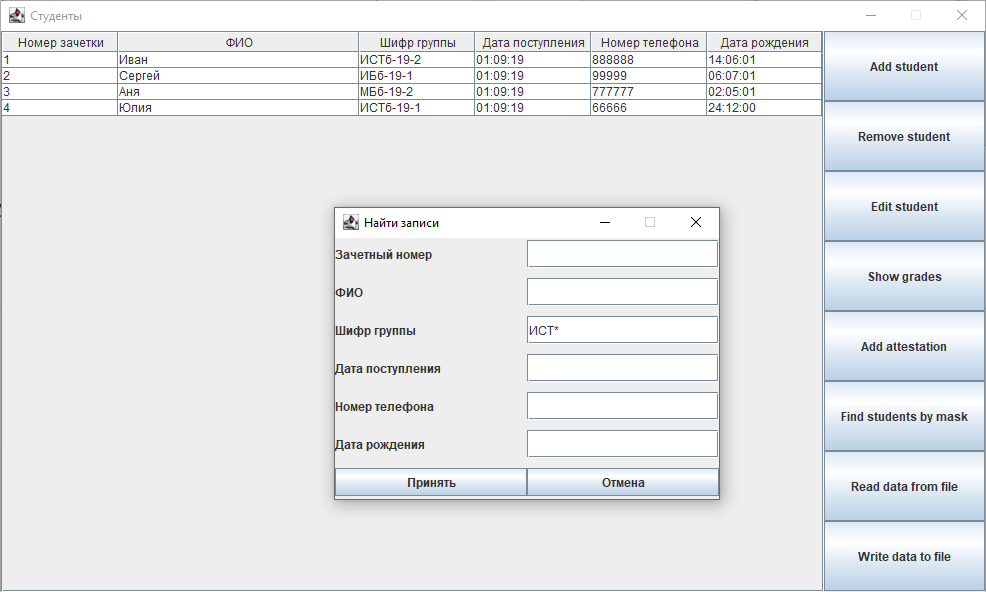


Рисунок 25 – Поиск студентов по маске

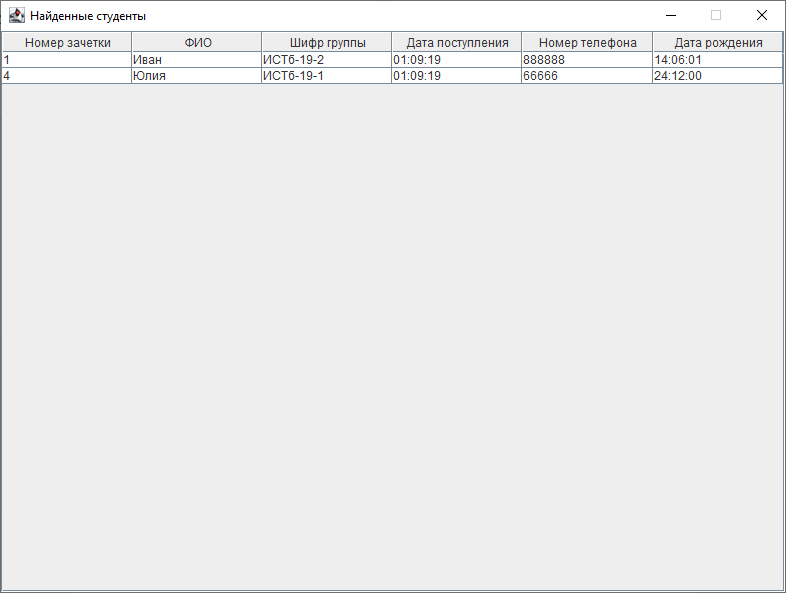


Рисунок 26 – Показ найденных студентов в таблице

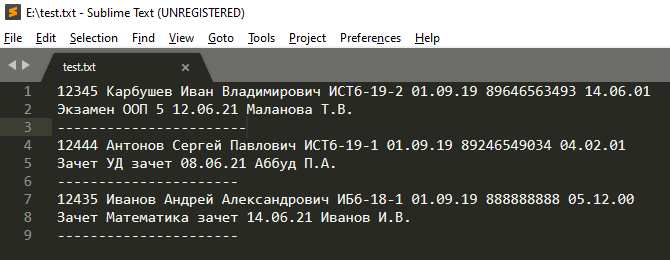


Рисунок 27 – Данные в файле для чтения

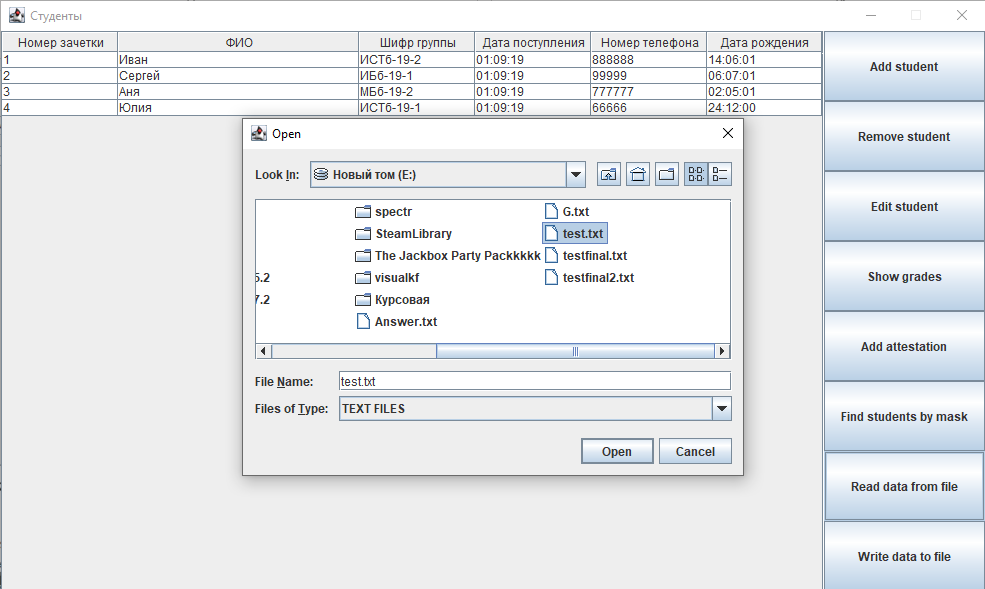


Рисунок 28 – Открытие файла для чтения

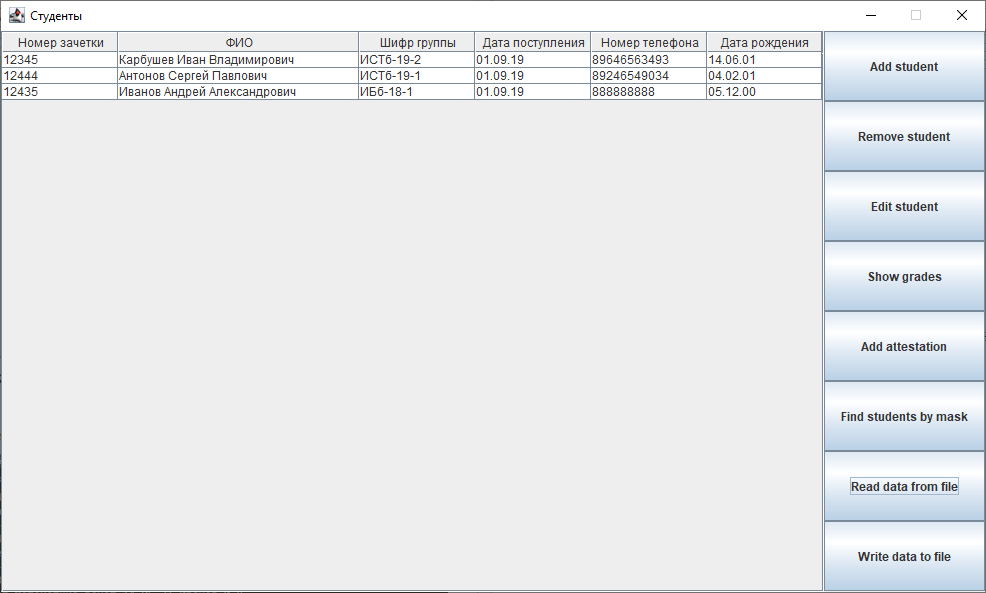


Рисунок 29 – Прочитаны данные с файла

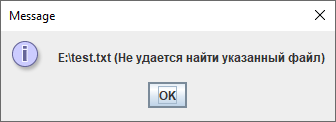


Рисунок 30 – Не удается найти файл при чтении

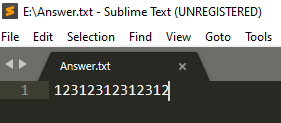


Рисунок 31 – Некорректные данные для чтения

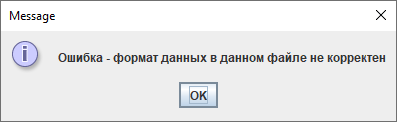


Рисунок 32 – Ошибка о формате данных

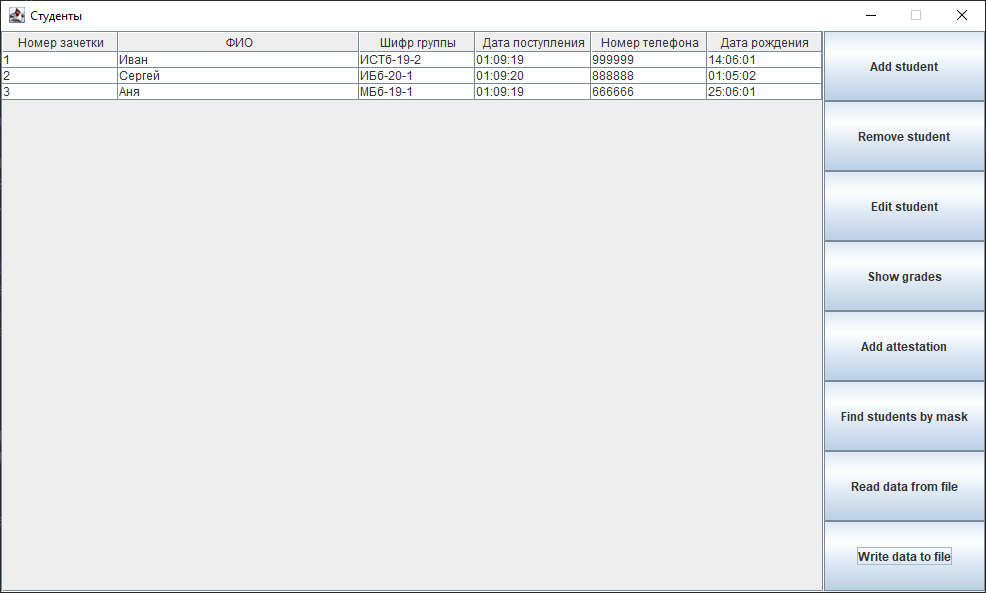


Рисунок 33 – Данные для записи в файл

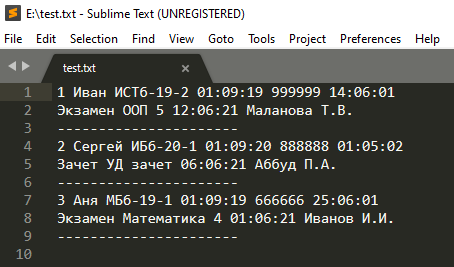


Рисунок 34 – Данные из таблицы записаны в файл

5 Листинг программы на Java

**Класс Main**

package Main;  
  
import View.WindowManager;  
  
public class Main {  
  
 public static void main(String[] args) {  
 new WindowManager();  
 }  
}

**Класс Attestation**

package Students.Attestation;  
  
public abstract class Attestation {  
 protected String subject;  
 protected String date;  
 protected String teacherName;  
  
 public Attestation(String subject, String date, String teacherName) {  
 this.subject = subject;  
 this.date = date;  
 this.teacherName = teacherName;  
 }  
  
 public String getSubject() {  
 return subject;  
 }  
  
 public String getDate() {  
 return this.date;  
 }  
  
 public String getTeacherName() {  
 return teacherName;  
 }  
  
 public void setSubject(String subject) {  
 this.subject = subject;  
 }  
  
 public void setTeacherName(String teacherName) {  
 this.teacherName = teacherName;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (o instanceof Attestation) {  
 return subject.equals(((Attestation) o).getSubject()) &&  
 date.equals(((Attestation) o).getDate()) &&  
 teacherName.equals(((Attestation) o).getTeacherName());  
 }  
 return false;  
 }  
}

**Класс Exam**

package Students.Attestation;  
  
public class Exam extends Attestation {  
 public Exam(String subject, String date, String teacherName) {  
 super(subject, date, teacherName);  
 }  
  
 @Override  
 public String toString() {  
 return "Экзамен";  
 }  
}

**Класс Test**

package Students.Attestation;  
  
public class Test extends Attestation {  
 public Test(String subject, String date, String teacherName) {  
 super(subject, date, teacherName);  
 }  
  
 @Override  
 public String toString() {  
 return "Зачет";  
 }  
}

**Класс Student**

package Students.Attestation;  
  
import java.util.HashMap;  
  
public class Student {  
 private final String name;  
 private final String enrolDate;  
 private final String birthDate;  
 private final HashMap<Attestation, String> sessionMarks = new HashMap<>();  
 private final String studentId;  
 private String phoneNumber;  
 private String group;  
  
  
 public Student(String studentId, String name, String group, String enrolDate, String phoneNumber, String birthDate) {  
 this.studentId = studentId;  
 this.name = name;  
 this.group = group;  
 this.enrolDate = enrolDate;  
 this.birthDate = birthDate;  
 this.phoneNumber = phoneNumber;  
 }  
  
 public void setMark(Attestation attestation, String mark) {  
 sessionMarks.put(attestation, mark);  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getGroup() {  
 return group;  
 }  
  
 public void setGroup(String group) {  
 this.group = group;  
 }  
  
 public String getStudentId() {  
 return studentId;  
 }  
  
 public String getEnrolDate() {  
 return enrolDate;  
 }  
  
 public String getBirthDate() {  
 return birthDate;  
 }  
  
 public String getPhoneNumber() {  
 return phoneNumber;  
 }  
  
 public void setPhoneNumber(String phoneNumber) {  
 this.phoneNumber = phoneNumber;  
 }  
  
 public HashMap<Attestation, String> getSessionMarks() {  
 return sessionMarks;  
 }  
  
 public void deleteAttestation(Attestation attestation) {  
 sessionMarks.remove(attestation);  
 }  
}

**Интерфейс StudentListManageAble**

package Students.Attestation;  
  
public interface StudentListManageable {  
 void add(int index, Attestation attestation, String mark);  
  
 void add(Student student);  
  
 void remove(int index);  
  
 int size();  
  
 Student get(int index);  
  
 void set(int index, Student student);  
  
 void clearAll();  
}

**Класс StudentManager**

package Students.Attestation;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class StudentManager implements StudentListManageable {  
 private final List<Student> studentList;  
  
 public StudentManager() {  
 studentList = new ArrayList<>();  
 }  
  
 @Override  
 public void add(int index, Attestation attestation, String mark) {  
 Student student = studentList.get(index);  
 student.setMark(attestation, mark);  
 }  
  
 @Override  
 public void add(Student student) {  
 studentList.add(student);  
 }  
   
 @Override  
 public void remove(int index) {  
 studentList.remove(index);  
 }  
  
 @Override  
 public int size() {  
 return studentList.size();  
 }  
  
 @Override  
 public Student get(int index) {  
 return studentList.get(index);  
 }  
  
 @Override  
 public void set(int index, Student student) {  
 studentList.set(index, student);  
 }  
  
 @Override  
 public void clearAll() {  
 studentList.clear();  
 }  
}

**Класс StudentFileManager**

package Files;  
  
import Students.Attestation.Attestation;  
import Students.Attestation.Exam;  
import Students.Attestation.Student;  
import Students.Attestation.Test;  
import View.StudentTableModel;  
  
import javax.swing.\*;  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
public class StudentFileManager {  
 public static void writeToFile(FileWriter fw, StudentTableModel model, JFrame frame) {  
 try {  
 //ТУТ НАПИСАТЬ ПОДЗАГОЛОВКИ ТОГО, ЧТО ВЫВОДИТСЯ  
 BufferedWriter bw = new BufferedWriter(fw);  
 for (int i = 0; i < model.getRowCount(); i++) {  
 Student stud = model.getStudent(i);  
 bw.write(stud.getStudentId() + " " +  
 stud.getName() + " " +  
 stud.getGroup() + " " +  
 stud.getEnrolDate() + " " +  
 stud.getPhoneNumber() + " " +  
 stud.getBirthDate() +  
 "\n");  
 List<Attestation> list = new ArrayList<>(stud.getSessionMarks().keySet());  
 for (var item : list) {  
 bw.write(item + " " +  
 item.getSubject() + " " +  
 stud.getSessionMarks().get(item) + " " +  
 item.getDate() + " " +  
 item.getTeacherName() +  
 "\n");  
 }  
 bw.write("----------------------\n");  
 }  
 bw.close();  
 } catch (IOException fileNotFoundException) {  
 fileNotFoundException.printStackTrace();  
 }  
 }  
  
 public static void readFromFile(FileReader fr, StudentTableModel model, JFrame frame) {  
 try {  
 model.clearAll();  
 BufferedReader br = new BufferedReader(fr);  
 String line;  
 int studInd = 0;  
 boolean studLine = true;  
 while ((line = br.readLine()) != null) {  
 if (line.charAt(0) == '-') {  
 studLine = true;  
 studInd++;  
 continue;  
 }  
 String[] studentData = line.split(" ");  
 if (studLine) {  
 for (int i = 0; i < model.getRowCount(); i++) {  
 Student s = model.getStudent(i);  
 }  
 model.add(new Student(  
 studentData[0],  
 studentData[1] + " " + studentData[2] + " " + studentData[3],  
 studentData[4],  
 studentData[5],  
 studentData[6],  
 studentData[7]  
 ));  
 studLine = false;  
 }  
 else {  
 Attestation at;  
 if (studentData[0].equals("экзамен")) at = new Exam(studentData[1], studentData[3], studentData[4]);  
 else at = new Test(studentData[1], studentData[3], studentData[4]);  
 model.add(studInd, at, studentData[2]);  
 }  
 }  
 } catch (IndexOutOfBoundsException ex) {  
 JOptionPane.showMessageDialog(frame, "Ошибка - формат данных в данном файле не корректен");  
 } catch (IOException ex) {  
 JOptionPane.showMessageDialog(frame, ex.getMessage());  
 }  
 }  
}

**Класс MarksTableModel**

package View;  
  
import Students.Attestation.Attestation;  
import Students.Attestation.Exam;  
import Students.Attestation.Student;  
  
import javax.swing.table.AbstractTableModel;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
  
public class MarksTableModel extends AbstractTableModel {  
 private final StudentTableModel mainModel;  
 private int index;  
 private int rows;  
  
 public MarksTableModel(StudentTableModel model) {  
 mainModel = model;  
 }  
  
 @Override  
 public int getRowCount() {  
 return rows;  
 }  
  
 @Override  
 public int getColumnCount() {  
 return 5;  
 }  
  
 @Override  
 public String getColumnName(int column) {  
 switch (column) {  
 case 0:  
 return "Предмет";  
 case 1:  
 return "Форма аттестации";  
 case 2:  
 return "Оценка";  
 case 3:  
 return "Преподаватель";  
 case 4:  
 return "Дата экзамена";  
 }  
 return super.getColumnName(column);  
 }  
  
 @Override  
 public Object getValueAt(int rowIndex, int columnIndex) {  
 Student st = mainModel.getStudent(index);  
 HashMap<Attestation, String> marks = st.getSessionMarks();  
 List<Attestation> keys = new ArrayList<>(marks.keySet());  
 Attestation at = keys.get(rowIndex);  
  
 switch (columnIndex) {  
 case 0:  
 return at.getSubject();  
 case 1:  
 return at instanceof Exam ? "экзамен" : "зачет";  
 case 2:  
 return marks.get(at);  
 case 3:  
 return at.getTeacherName();  
 case 4:  
 return at.getDate();  
 }  
 return null;  
 }  
  
 public void setIndex(int index) {  
 this.index = index;  
 this.fireTableDataChanged();  
 }  
  
 public void setRows(int rows) {  
 this.rows = rows;  
 }  
}

**Класс StudentTableModel**

package View;  
  
import Students.Attestation.Attestation;  
import Students.Attestation.Student;  
import Students.Attestation.StudentManager;  
  
import javax.swing.table.AbstractTableModel;  
  
public class StudentTableModel extends AbstractTableModel {  
 private final StudentManager studentList;  
  
 public StudentTableModel() {  
 studentList = new StudentManager();  
 }  
  
 @Override  
 public int getRowCount() {  
 return studentList.size();  
 }  
  
 @Override  
 public int getColumnCount() {  
 return 6;  
 }  
  
 @Override  
 public String getColumnName(int column) {  
 switch (column) {  
 case 0:  
 //номер зачетки  
 return "Номер зачетки";  
 case 1:  
 //ФИО  
 return "ФИО";  
 case 2:  
 //шифр группы  
 return "Шифр группы";  
 case 3:  
 //дата поступления  
 return "Дата поступления";  
 case 4:  
 //номер телефона  
 return "Номер телефона";  
 case 5:  
 //дата рождения  
 return "Дата рождения";  
 }  
 return super.getColumnName(column);  
 }  
  
 @Override  
 public Object getValueAt(int rowIndex, int columnIndex) {  
 Student student = studentList.get(rowIndex);  
 switch (columnIndex) {  
 case 0:  
 //номер зачетки  
 return student.getStudentId();  
 case 1:  
 //ФИО  
 return student.getName();  
 case 2:  
 //шифр группы  
 return student.getGroup();  
 case 3:  
 //дата поступления  
 return student.getEnrolDate();  
 case 4:  
 //номер телефона  
 return student.getPhoneNumber();  
 case 5:  
 //дата рождения  
 return student.getBirthDate();  
 }  
 return null;  
 }  
  
 @Override  
 public boolean isCellEditable(int rowIndex, int columnIndex) {  
 return super.isCellEditable(rowIndex, columnIndex);  
 }  
  
 @Override  
 public void setValueAt(Object aValue, int rowIndex, int columnIndex) {  
 super.setValueAt(aValue, rowIndex, columnIndex);  
 }  
  
 public Student getStudent(int index) {  
 return studentList.get(index);  
 }  
  
 public void add(Student student) {  
 studentList.add(student);  
 this.fireTableDataChanged();  
 }  
  
 public void remove(int index) {  
 studentList.remove(index);  
 this.fireTableDataChanged();  
 }  
  
 public void edit(int index, Student student) {  
 studentList.set(index, student);  
 this.fireTableDataChanged();  
 }  
  
 public void add(int index, Attestation attestation, String mark) {  
 studentList.add(index, attestation, mark);  
 }  
  
 public int getMarksCount(int index) {  
 return studentList.get(index).getSessionMarks().size();  
 }  
  
 public void clearAll() {  
 studentList.clearAll();  
 fireTableDataChanged();  
 }  
}

**Класс WindowManager**

package View;  
  
import Files.StudentFileManager;  
import Students.Attestation.Attestation;  
import Students.Attestation.Exam;  
import Students.Attestation.Student;  
import Students.Attestation.Test;  
import javax.swing.\*;  
import javax.swing.filechooser.FileNameExtensionFilter;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.io.\*;  
import java.util.\*;  
import java.util.List;  
  
  
public class WindowManager extends JFrame {  
 JTable table, markTable;  
 JFrame frame;  
 JPanel panel, rightPanel;  
 JButton addButton, removeButton, editButton, gradesButton, addAtButton, showAtButton, readButton, writeButton;  
 JScrollPane scrollPane;  
 StudentTableModel model;  
 MarksTableModel markModel;  
  
 public WindowManager() {  
 init();  
 }  
  
 private void init() {  
 frame = new JFrame();  
 setTitle("Студенты");  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
 panel = new JPanel();  
 panel.setLayout(new BorderLayout());  
 rightPanel = new JPanel();  
 rightPanel.setLayout(new GridLayout(8, 1));  
  
 addButton = new JButton("Add student");  
 addButton.setPreferredSize(new Dimension(121, 50));  
 addButton.addActionListener(new AddButtonListener());  
 removeButton = new JButton("Remove student");  
 removeButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 int row = table.getSelectedRow();  
 if (row == -1) {  
 JOptionPane.*showMessageDialog*(frame, "Вы не выбрали запись");  
 return;  
 }  
 model.remove(row);  
 }  
 });  
 editButton = new JButton("Edit student");  
 editButton.addActionListener(new EditButtonListener());  
 gradesButton = new JButton("Show grades");  
 gradesButton.addActionListener(new ShowMarksButtonListener());  
 addAtButton = new JButton("Add attestation");  
 addAtButton.addActionListener(new AddAttestationButtonListener());  
 showAtButton = new JButton("Find students by mask");  
 showAtButton.addActionListener(new SearchButtonListener());  
 readButton = new JButton("Read data from file");  
 readButton.addActionListener(new ReadFileSelector());  
 writeButton = new JButton("Write data to file");  
 writeButton.addActionListener(new WriteFileSelector());  
  
 //добавление кнопок  
 rightPanel.add(addButton);  
 rightPanel.add(removeButton);  
 rightPanel.add(editButton);  
 rightPanel.add(gradesButton);  
 rightPanel.add(addAtButton);  
 rightPanel.add(showAtButton);  
 rightPanel.add(readButton);  
 rightPanel.add(writeButton);  
  
 model = new StudentTableModel();  
 table = new JTable(model);  
 table.getTableHeader().setReorderingAllowed(false);  
 table.getColumnModel().getColumn(1).setMaxWidth(Integer.*MAX\_VALUE*);  
 table.getColumnModel().getColumn(1).setMinWidth(200);  
 scrollPane = new JScrollPane(table);  
  
 markModel = new MarksTableModel(model);  
 markTable = new JTable(markModel);  
  
 panel.add(scrollPane, BorderLayout.*CENTER*);  
 panel.add(rightPanel, BorderLayout.*EAST*);  
  
 setContentPane(panel);  
 setPreferredSize(new Dimension(1000, 600));  
 pack();  
 setResizable(false);  
 setLocationRelativeTo(null);  
 setVisible(true);  
 }  
  
 private class AddButtonListener extends JFrame implements ActionListener {  
 JTextField personalId, name, groupId, enrolDate, mobileNumber, birthDate;  
 JLabel idL, nameL, groupIdL, enrolL, phoneL, birthL;  
 JButton buttonApply, buttonCancel;  
  
 public void fieldsActionsInit() {  
 personalId.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 buttonApply.doClick();  
 }  
 });  
 name.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 buttonApply.doClick();  
 }  
 });  
 groupId.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 buttonApply.doClick();  
 }  
 });  
 enrolDate.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 buttonApply.doClick();  
 }  
 });  
 mobileNumber.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 buttonApply.doClick();  
 }  
 });  
 birthDate.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 buttonApply.doClick();  
 }  
 });  
 }  
  
 public void addingComponentsToPanel() {  
 GridLayout grid = new GridLayout(7, 2);  
 grid.setVgap(10);  
 JPanel addPanel = new JPanel(grid);  
 addPanel.add(idL);  
 addPanel.add(personalId);  
 addPanel.add(nameL);  
 addPanel.add(name);  
 addPanel.add(groupIdL);  
 addPanel.add(groupId);  
 addPanel.add(enrolL);  
 addPanel.add(enrolDate);  
 addPanel.add(phoneL);  
 addPanel.add(mobileNumber);  
 addPanel.add(birthL);  
 addPanel.add(birthDate);  
 addPanel.add(buttonApply);  
 addPanel.add(buttonCancel);  
 setContentPane(addPanel);  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 setTitle("Добавить запись");  
  
 idL = new JLabel("Зачетный номер");  
 nameL = new JLabel("ФИО");  
 groupIdL = new JLabel("Шифр группы");  
 enrolL = new JLabel("Дата поступления");  
 phoneL = new JLabel("Номер телефона");  
 birthL = new JLabel("Дата рождения");  
  
  
 personalId = new JTextField(10);  
 name = new JTextField(40);  
 groupId = new JTextField(8);  
 enrolDate = new JTextField(8);  
 enrolDate.addKeyListener(new DateKeyAdapter());  
 mobileNumber = new JTextField(11);  
 birthDate = new JTextField(8);  
 birthDate.addKeyListener(new DateKeyAdapter());  
 fieldsActionsInit();  
  
 buttonApply = new JButton("Принять");  
 buttonApply.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (personalId.getText().equals("") ||  
 name.getText().equals("") ||  
 groupId.getText().equals("") ||  
 enrolDate.getText().equals("") ||  
 mobileNumber.getText().equals("") ||  
 birthDate.getText().equals(""))  
 {  
 JOptionPane.*showMessageDialog*(frame, "Не все поля заполнены");  
 return;  
 }  
 for (var item : name.getText().toCharArray()) {  
 if (isNumber(item) != -1) {  
 JOptionPane.*showMessageDialog*(frame, "В поле имени или названия введены числа");  
 return;  
 }  
 }  
  
 Student student = new Student(personalId.getText(),  
 name.getText(),  
 groupId.getText(),  
 enrolDate.getText(),  
 mobileNumber.getText(),  
 birthDate.getText());  
 model.add(student);  
 dispose();  
 }  
 });  
 buttonCancel = new JButton("Отмена");  
 buttonCancel.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 dispose();  
 }  
 });  
  
 addingComponentsToPanel();  
 setSize(400, 300);  
 setResizable(false);  
 setVisible(true);  
 }  
 }  
  
 private class EditButtonListener extends JFrame implements ActionListener {  
 JTextField personalId, name, groupId, enrolDate, mobileNumber, birthDate;  
 JLabel idL, nameL, groupIdL, enrolL, phoneL, birthL;  
 JButton buttonApply, buttonCancel;  
 int row;  
  
 public void addingComponentsToPanel() {  
 GridLayout grid = new GridLayout(7, 2);  
 grid.setVgap(10);  
 JPanel editPanel = new JPanel(grid);  
 editPanel.add(idL);  
 editPanel.add(personalId);  
 editPanel.add(nameL);  
 editPanel.add(name);  
 editPanel.add(groupIdL);  
 editPanel.add(groupId);  
 editPanel.add(enrolL);  
 editPanel.add(enrolDate);  
 editPanel.add(phoneL);  
 editPanel.add(mobileNumber);  
 editPanel.add(birthL);  
 editPanel.add(birthDate);  
 editPanel.add(buttonApply);  
 editPanel.add(buttonCancel);  
  
 setContentPane(editPanel);  
 }  
  
 public void fieldsActionsInit() {  
 personalId.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 buttonApply.doClick();  
 }  
 });  
 name.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 buttonApply.doClick();  
 }  
 });  
 groupId.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 buttonApply.doClick();  
 }  
 });  
 enrolDate.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 buttonApply.doClick();  
 }  
 });  
 mobileNumber.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 buttonApply.doClick();  
 }  
 });  
 birthDate.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 buttonApply.doClick();  
 }  
 });  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 row = table.getSelectedRow();  
 if (row == -1) {  
 JOptionPane.*showMessageDialog*(frame, "Вы не выбрали запись");  
 return;  
 }  
  
 setTitle("Редактировать запись");  
  
 idL = new JLabel("Зачетный номер");  
 nameL = new JLabel("ФИО");  
 groupIdL = new JLabel("Шифр группы");  
 enrolL = new JLabel("Дата поступления");  
 phoneL = new JLabel("Номер телефона");  
 birthL = new JLabel("Дата рождения");  
  
  
 //поля ввода  
 personalId = new JTextField(10);  
 personalId.setText((String) model.getValueAt(row, 0));  
 name = new JTextField(40);  
 name.setText((String) model.getValueAt(row, 1));  
 groupId = new JTextField(8);  
 groupId.setText((String) model.getValueAt(row, 2));  
 enrolDate = new JTextField(8);  
 enrolDate.setText((String) model.getValueAt(row, 3));  
 enrolDate.addKeyListener(new DateKeyAdapter());  
 mobileNumber = new JTextField(11);  
 mobileNumber.setText((String) model.getValueAt(row, 4));  
 birthDate = new JTextField(8);  
 birthDate.setText((String) model.getValueAt(row, 5));  
 birthDate.addKeyListener(new DateKeyAdapter());  
 fieldsActionsInit();  
  
 buttonApply = new JButton("Принять");  
 buttonApply.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 for (var item : name.getText().toCharArray()) {  
 if (isNumber(item) != -1) {  
 JOptionPane.*showMessageDialog*(frame, "В поле имени или названия введены числа");  
 return;  
 }  
 }  
 Student student = new Student(personalId.getText(),  
 name.getText(),  
 groupId.getText(),  
 enrolDate.getText(),  
 mobileNumber.getText(),  
 birthDate.getText());  
 model.edit(row, student);  
 dispose();  
 }  
 });  
 buttonCancel = new JButton("Отмена");  
 buttonCancel.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 dispose();  
 }  
 });  
  
 addingComponentsToPanel();  
 setSize(400, 300);  
 setResizable(false);  
 setVisible(true);  
 }  
 }  
  
 private class AddAttestationButtonListener extends JFrame implements ActionListener {  
 JLabel at, sub, gr, da, teacher;  
 JTextField subject, grade, date, teacherName;  
 JComboBox<String> attestationType;  
 JPanel aaPanel;  
 JButton apply, cancel;  
 int row;  
  
 public void addingComponentsToPanel() {  
 GridLayout grid = new GridLayout(6, 2);  
 grid.setVgap(10);  
 aaPanel = new JPanel(grid);  
 aaPanel.add(at);  
 aaPanel.add(attestationType);  
 aaPanel.add(sub);  
 aaPanel.add(subject);  
 aaPanel.add(gr);  
 aaPanel.add(grade);  
 aaPanel.add(da);  
 aaPanel.add(date);  
 aaPanel.add(teacher);  
 aaPanel.add(teacherName);  
 aaPanel.add(apply);  
 aaPanel.add(cancel);  
  
 setContentPane(aaPanel);  
 }  
  
 public void fieldsActionsInit() {  
 subject.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 apply.doClick();  
 }  
 });  
 grade.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 apply.doClick();  
 }  
 });  
 date.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 apply.doClick();  
 }  
 });  
 teacherName.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 apply.doClick();  
 }  
 });  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 setTitle("Добавить аттестацию");  
 row = table.getSelectedRow();  
 if (row == -1) {  
 JOptionPane.*showMessageDialog*(frame, "Вы не выбрали запись");  
 return;  
 }  
  
 attestationType = new JComboBox<>(new String[]{"экзамен", "зачет"});  
 attestationType.setSelectedItem("экзамен");  
 grade = new JTextField(5);  
 subject = new JTextField(30);  
 date = new JTextField(8);  
 date.addKeyListener(new DateKeyAdapter());  
 teacherName = new JTextField(40);  
 fieldsActionsInit();  
  
 at = new JLabel("Форма аттестации");  
 sub = new JLabel("Предмет");  
 gr = new JLabel("Оценка");  
 da = new JLabel("Дата");  
 teacher = new JLabel("Преподаватель");  
  
 apply = new JButton("Принять");  
 cancel = new JButton("Отмена");  
 apply.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (grade.getText().equals("") ||  
 subject.getText().equals("") ||  
 date.getText().equals("") ||  
 teacherName.getText().equals(""))  
 {  
 JOptionPane.*showMessageDialog*(frame, "Вы не выбрали запись");  
 return;  
 }  
 for (var item : subject.getText().toCharArray()) {  
 if (isNumber(item) != -1) {  
 JOptionPane.*showMessageDialog*(frame, "В поле имени или названия введены числа");  
 return;  
 }  
 }  
 for (var item : teacherName.getText().toCharArray()) {  
 if (isNumber(item) != -1) {  
 JOptionPane.*showMessageDialog*(frame, "В поле имени или названия введены числа");  
 return;  
 }  
 }  
  
 Attestation attestation;  
 if (attestationType.getSelectedItem().equals("экзамен")) {  
 attestation = new Exam(subject.getText(), date.getText(), teacherName.getText());  
 }  
 else {  
 attestation = new Test(subject.getText(), date.getText(), teacherName.getText());  
 }  
 model.add(row, attestation, grade.getText());  
 dispose();  
 }  
 });  
 cancel.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 dispose();  
 }  
 });  
  
 addingComponentsToPanel();  
 setSize(400, 300);  
 setResizable(false);  
 setVisible(true);  
 }  
 }  
  
 private class ShowMarksButtonListener extends JFrame implements ActionListener {  
 int row;  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 row = table.getSelectedRow();  
 if (row == -1) {  
 JOptionPane.*showMessageDialog*(frame, "Вы не выбрали запись");  
 return;  
 }  
  
 setContentPane(new JScrollPane(markTable));  
 markModel.setRows(model.getMarksCount(row));  
 markModel.setIndex(row);  
 pack();  
  
 setResizable(false);  
 setLocationRelativeTo(null);  
 setTitle("Оценки");  
 setSize(600, 600);  
 setVisible(true);  
 }  
 }  
  
 private class SearchButtonListener extends JFrame implements ActionListener {  
 JTextField id, name, group, enrolDate, phoneNumber, birthDate;  
 JLabel idL, nameL, groupIdL, enrolL, phoneL, birthL;  
 JButton apply, cancel;  
 List<Integer> rows = new ArrayList<>();  
 JTable foundTable;  
  
 public boolean checkOccurrence(String tableData, String mask) {  
 if (mask.equals("")) return true;  
 tableData = tableData.toLowerCase();  
 mask = mask.toLowerCase();  
 boolean isFirstStar = mask.charAt(0) == '\*';  
 boolean isLastStar = mask.charAt(mask.length()-1) == '\*';  
 String[] arr = mask.split("\\\*");  
 int sumOfMask = 0;  
 for (String item : arr) {  
 sumOfMask += item.length();  
 }  
 if (tableData.length() < sumOfMask) return false;  
 if (arr.length == 1 && !mask.contains("\*")) {  
 return tableData.equals(mask);  
 }  
 if (isLastStar) {  
 for (int i = 0; i < arr[0].length(); i++) {  
 if (tableData.charAt(i) != arr[0].charAt(i)) {  
 System.*out*.println(tableData.charAt(i) + " " + arr[0].charAt(i));  
 return false;  
 }  
 }  
 }  
 if (isFirstStar) {  
 for (int i = tableData.length()-1, j = arr[arr.length-1].length()-1;  
 i > tableData.length()-1-arr[arr.length-1].length();  
 i--, j--)  
 {  
 if (tableData.charAt(i) != arr[arr.length-1].charAt(j)) return false;  
 }  
 }  
 for (var item : arr) {  
 if (tableData.contains(item)) {  
 tableData = tableData.substring(tableData.indexOf(item) + item.length());  
 } else {  
 return false;  
 }  
 }  
 return true;  
 }  
  
 private void showResultOfSearch() {  
 StudentTableModel foundStudents = new StudentTableModel();  
 foundStudents.clearAll();  
 for (int number : rows) {  
 foundStudents.add(model.getStudent(number));  
 }  
 foundTable = new JTable(foundStudents);  
 JFrame tableFrame = new JFrame("Найденные студенты");  
 tableFrame.setContentPane(new JScrollPane(foundTable));  
 tableFrame.setSize(800, 600);  
 tableFrame.setResizable(false);  
 tableFrame.setLocationRelativeTo(null);  
 tableFrame.setVisible(true);  
 }  
  
 private void addingComponentsToPanel() {  
 GridLayout grid = new GridLayout(7, 2);  
 grid.setVgap(10);  
 JPanel searchPanel = new JPanel(grid);  
 searchPanel.add(idL);  
 searchPanel.add(id);  
 searchPanel.add(nameL);  
 searchPanel.add(name);  
 searchPanel.add(groupIdL);  
 searchPanel.add(group);  
 searchPanel.add(enrolL);  
 searchPanel.add(enrolDate);  
 searchPanel.add(phoneL);  
 searchPanel.add(phoneNumber);  
 searchPanel.add(birthL);  
 searchPanel.add(birthDate);  
 searchPanel.add(apply);  
 searchPanel.add(cancel);  
 setContentPane(searchPanel);  
 }  
  
 public void fieldsActionsInit() {  
 id.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 apply.doClick();  
 }  
 });  
 name.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 apply.doClick();  
 }  
 });  
 group.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 apply.doClick();  
 }  
 });  
 enrolDate.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 apply.doClick();  
 }  
 });  
 phoneNumber.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 apply.doClick();  
 }  
 });  
 birthDate.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 apply.doClick();  
 }  
 });  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (model.getRowCount() == 0) {  
 JOptionPane.*showMessageDialog*(frame, "В таблице нет записей");  
 return;  
 }  
  
 setTitle("Найти записи");  
  
 idL = new JLabel("Зачетный номер");  
 nameL = new JLabel("ФИО");  
 groupIdL = new JLabel("Шифр группы");  
 enrolL = new JLabel("Дата поступления");  
 phoneL = new JLabel("Номер телефона");  
 birthL = new JLabel("Дата рождения");  
  
 id = new JTextField();  
 name = new JTextField();  
 group = new JTextField();  
 enrolDate = new JTextField();  
 phoneNumber = new JTextField();  
 birthDate = new JTextField();  
  
 apply = new JButton("Принять");  
 apply.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String t1T = id.getText();  
 String t2T = name.getText();  
 String t3T = group.getText();  
 String t4T = enrolDate.getText();  
 String t5T = phoneNumber.getText();  
 String t6T = birthDate.getText();  
 rows.clear();  
  
 for (int i = 0; i < model.getRowCount(); i++) {  
 Student student = model.getStudent(i);  
 if (checkOccurrence(student.getStudentId(), t1T) &&  
 checkOccurrence(student.getName(), t2T) &&  
 checkOccurrence(student.getGroup(), t3T) &&  
 checkOccurrence(student.getEnrolDate(), t4T) &&  
 checkOccurrence(student.getPhoneNumber(), t5T) &&  
 checkOccurrence(student.getBirthDate(), t6T))  
 {  
 rows.add(i);  
 }  
 }  
 showResultOfSearch();  
 dispose();  
 }  
 });  
  
 cancel = new JButton("Отмена");  
 cancel.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 dispose();  
 }  
 });  
  
 fieldsActionsInit();  
 addingComponentsToPanel();  
 setSize(400, 300);  
 setResizable(false);  
 setVisible(true);  
 }  
 }  
  
 private class WriteFileSelector implements ActionListener {  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 FileNameExtensionFilter filter = new FileNameExtensionFilter("TEXT FILES", "txt", "text");  
 JFileChooser fc = new JFileChooser();  
 fc.setFileFilter(filter);  
 if (fc.showSaveDialog(null) == JFileChooser.*APPROVE\_OPTION*) {  
 try (FileWriter fw = new FileWriter(fc.getSelectedFile())) {  
 StudentFileManager.*writeToFile*(fw, model, frame);  
 } catch (IOException ex) {  
 JOptionPane.*showMessageDialog*(frame, ex.getMessage());  
 }  
 }  
 }  
 }  
  
 private class ReadFileSelector implements ActionListener {  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 FileNameExtensionFilter filter = new FileNameExtensionFilter("TEXT FILES", "txt", "text");  
 JFileChooser fc = new JFileChooser();  
 fc.setFileFilter(filter);  
 if (fc.showOpenDialog(null) == JFileChooser.*APPROVE\_OPTION*) {  
 try (FileReader fr = new FileReader(fc.getSelectedFile())) {  
 StudentFileManager.*readFromFile*(fr, model, frame);  
 } catch (IOException ex) {  
 JOptionPane.*showMessageDialog*(frame, ex.getMessage());  
 }  
 }  
 }  
 }  
  
 private class DateKeyAdapter extends KeyAdapter {  
 @Override  
 public void keyTyped(KeyEvent e) {  
 if (((JTextField)e.getSource()).getText().length() > 7) ((JTextField)e.getSource()).setText(((JTextField)e.getSource()).getText().substring(0, 7));  
 }  
  
 @Override  
 public void keyPressed(KeyEvent e) {  
 if ((((JTextField)e.getSource()).getText().length() == 2 || ((JTextField)e.getSource()).getText().length() == 5)  
 && e.getKeyChar() != KeyEvent.*VK\_BACK\_SPACE*) {  
 ((JTextField)e.getSource()).setText(((JTextField)e.getSource()).getText() + ":");  
 }  
 }  
  
 @Override  
 public void keyReleased(KeyEvent e) {  
 if (((JTextField)e.getSource()).getText().equals("")) return;  
 String[] arr = ((JTextField)e.getSource()).getText().split(":");  
 for (var item : arr) {  
 if (isNumber(item) == -1 && e.getKeyCode() != KeyEvent.*VK\_ENTER*) {  
 JOptionPane.*showMessageDialog*(frame, "В дате введены не числа");  
 return;  
 }  
 }  
 if (arr.length >= 2 && arr[1].length() == 2) {  
 int monthNumber = isNumber(arr[1]);  
 int dayNumber = isNumber(arr[0]);  
  
 if (monthNumber > 12) {  
 ((JTextField)e.getSource()).setText(((JTextField)e.getSource()).getText().substring(0,3) + "12" + ((JTextField)e.getSource()).getText().substring(5));  
 }  
 else if (monthNumber == 0) {  
 ((JTextField)e.getSource()).setText(((JTextField)e.getSource()).getText().substring(0,3) + "01" + ((JTextField)e.getSource()).getText().substring(5));  
 }  
  
 if (dayNumber == 0) {  
 ((JTextField)e.getSource()).setText("01" + ((JTextField)e.getSource()).getText().substring(2));  
 }  
  
 if (monthNumber == 2 && dayNumber > 28) {  
 ((JTextField)e.getSource()).setText("28" + ((JTextField)e.getSource()).getText().substring(2));  
 }  
 else if ((monthNumber == 4 || monthNumber == 6 || monthNumber == 9 || monthNumber == 11) && dayNumber > 30) {  
 ((JTextField)e.getSource()).setText("30" + ((JTextField)e.getSource()).getText().substring(2));  
 }  
 else if (dayNumber > 31) {  
 ((JTextField)e.getSource()).setText("31" + ((JTextField)e.getSource()).getText().substring(2));  
 }  
 }  
 }  
 }  
  
 private int isNumber(String stringToCheck) {  
 int result;  
 try {  
 result = Integer.*parseInt*(stringToCheck);  
 } catch (NumberFormatException ex) {  
 result = -1;  
 }  
 return result;  
 }  
  
 private int isNumber(char charToCheck) {  
 if (charToCheck < 48 || charToCheck > 57) return -1;  
 return charToCheck;  
 }  
}

6 Список литературы

1. Java. Экспресс-курс [электронный ресурс] // Сайт Александра Климова [сайт], URL: http://developer.alexanderklimov.ru/android/java/java.php (дата обращения: 11.01.2018)

2. API Specification for the Java 7 SE. [официальный сайт] URL: http://docs.oracle.com/javase/7/docs/api/ (дата обращения: 11.01.2018)

3. The Java Tutorials. SE [электронный ресурс], URL: http://docs.oracle.com/javase/tutorial/index.html (дата обращения: 11.01.2018)

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, -720 с.

5. Хабибуллин И.Ш. Java 7: для программистов / И. Ш. Хабибуллин. – Санкт-Петербург : БХВ–Петербург, 2014. – 768 с.

6. Файн Я. – Программирование на Java для детей, родителей, дедушек и бабушек, 2011.

7. 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.

8. Отношения классов — от UML к коду [Электронный ресурс] // Хабра-хабр: [сайт].

Питер, 2012. – 395 с.

8. Отношения классов — от UML к коду [Электронный ресурс] // Хабра-хабр: [сайт].