**Министерство образования и науки Российской Федерации**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ бюджетное образовательное учреждение высшего профессионального образования**

**«Санкт-Петербургский государственный электротехнический университет**

**“ЛЭТИ” им.В.И.Ульянова (Ленина)»**

**КАФЕДРА МОЭВМ**

**ОТЧЕТ**

**по лабораторно-практической работе № 5**

**«Сохранение и загрузка данных из файла»**

**по дисциплине «Объектно - ориентированное программирование на языке Java»**

Выполнил: Барченков П. А.

Факультет: КТИ

Группа: №3312

Подпись преподавателя: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2024

**Содержание**

[Цель работы 3](#_Toc181362490)

[Распечатки содержимого файлов с данными до и после внесения изменений 3](#_Toc181362491)

[Скриншоты, иллюстрирующие процесс загрузки данных в файл и выгрузки из него 5](#_Toc181362492)

[Текст программы 8](#_Toc181362493)

[Приложение 16](#_Toc181362494)

# Цель работы

Знакомство с организацией обмена данными между объектами экранной формы и файлом.

# Распечатки содержимого файлов с данными до и после внесения изменений

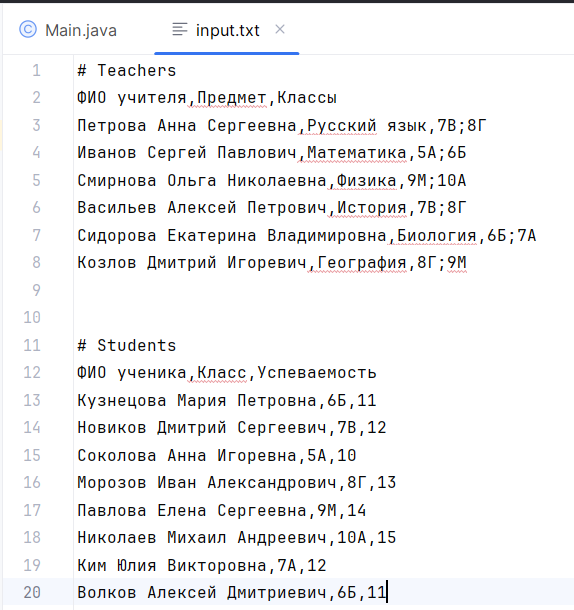


Рисунок 1 – Содержимое исходного файла

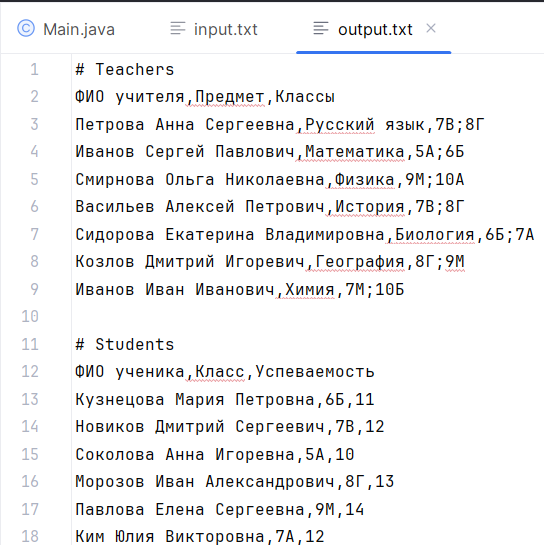


Рисунок 2 – Содержимое файла с данными после изменений

# Скриншоты, иллюстрирующие процесс загрузки данных в файл и выгрузки из него

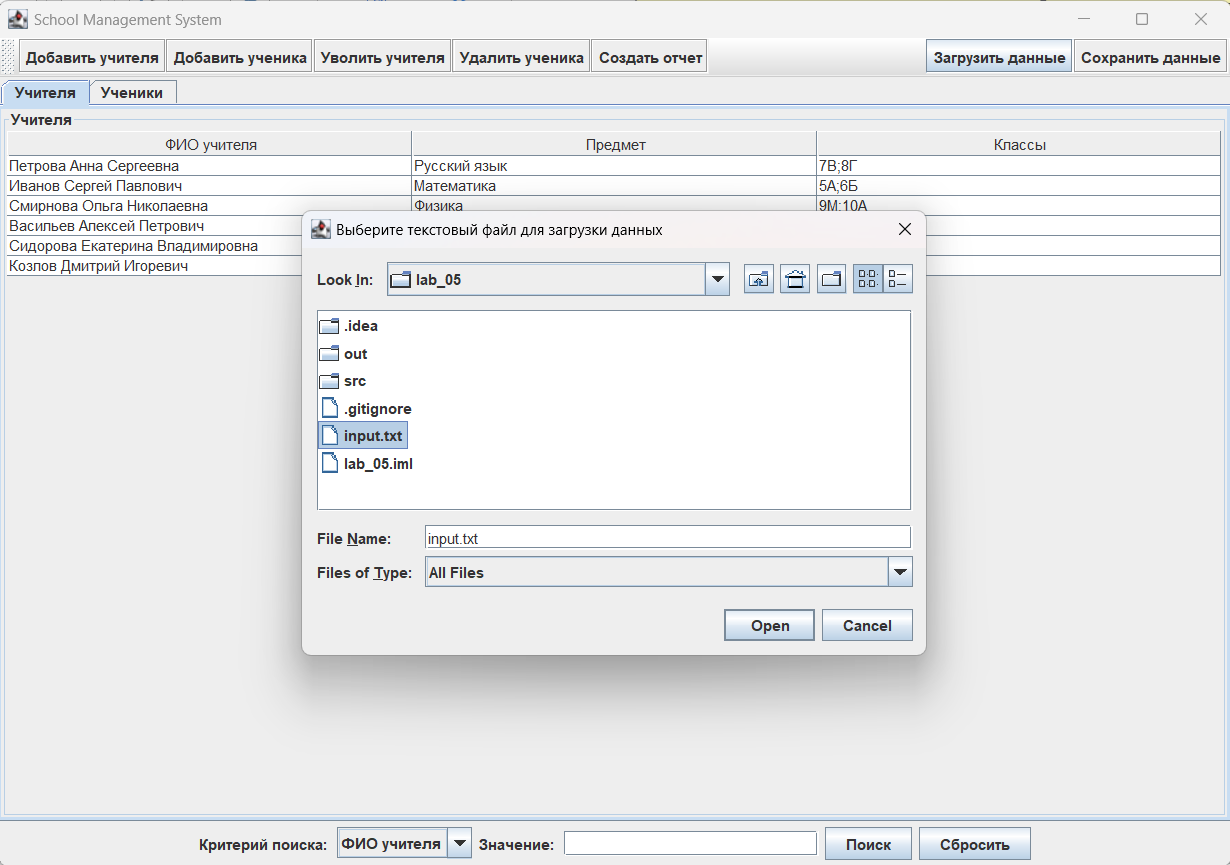


Рисунок 3 – Загрузка данных из исходного файла

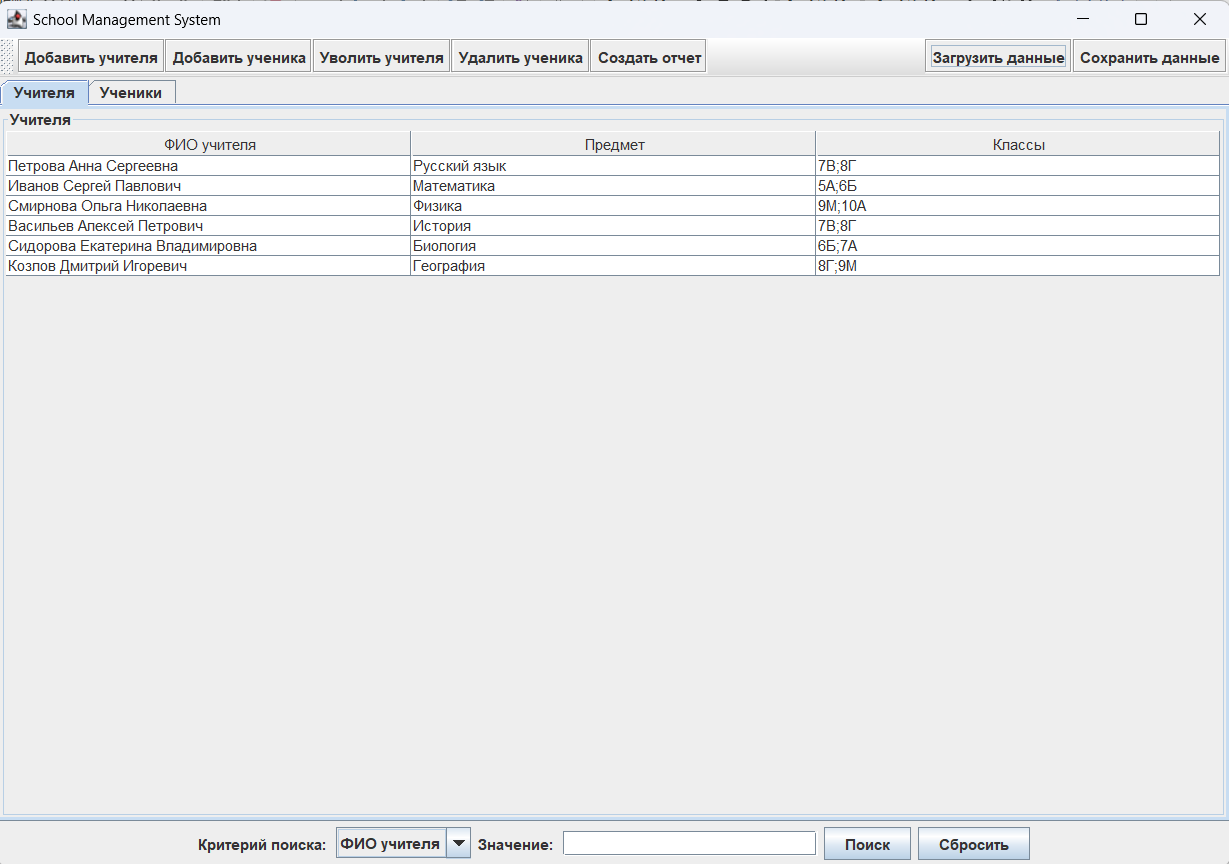


Рисунок 4 – Успешная загрузка данных из файла *input.txt*

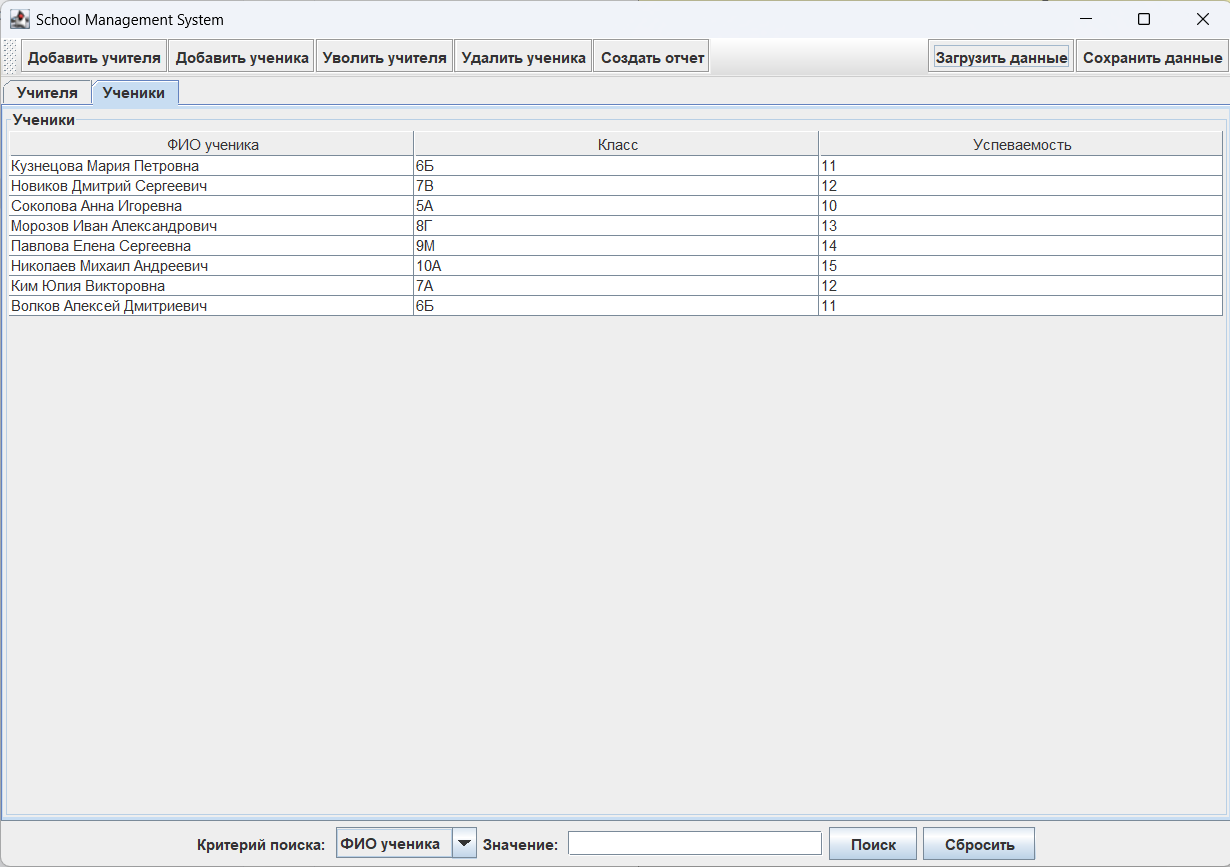


Рисунок 5 – Успешная загрузка данных из файла *input.txt*

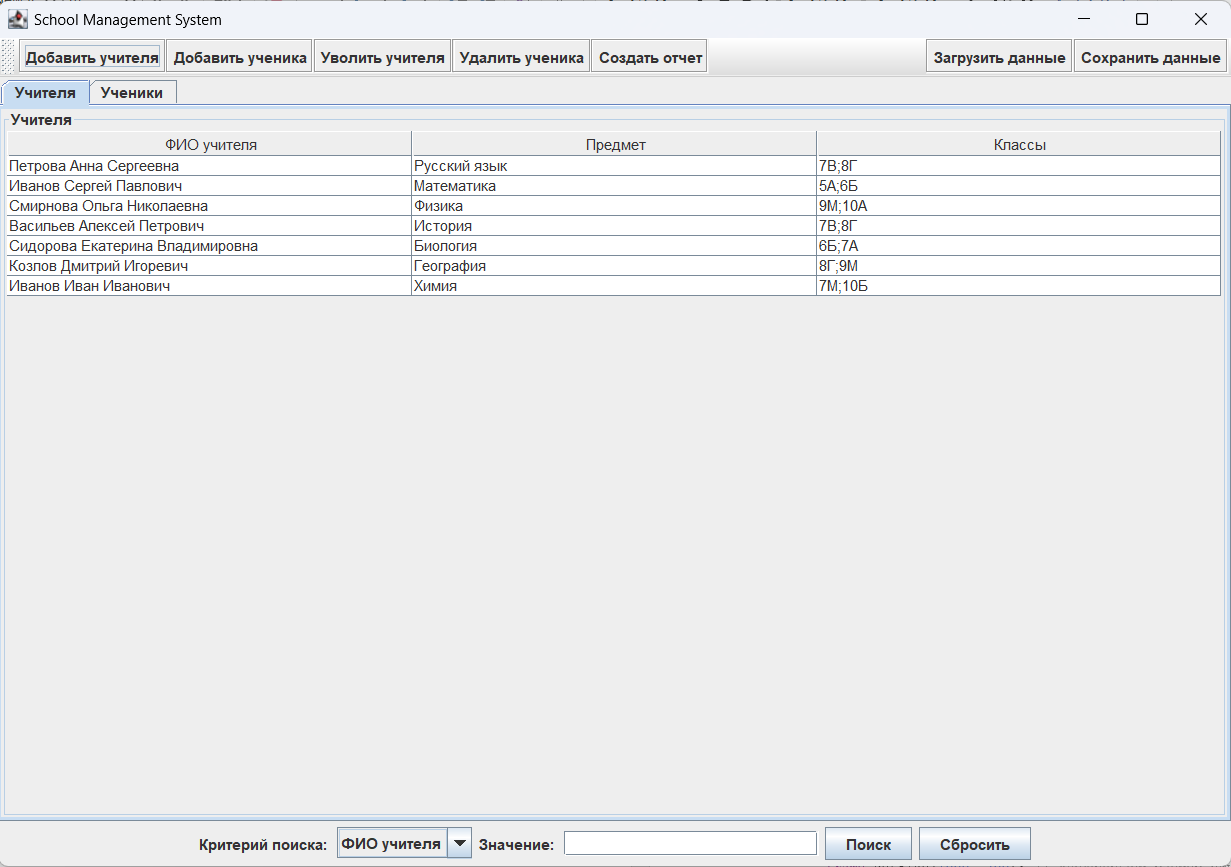


Рисунок 6 – Добавили учителя

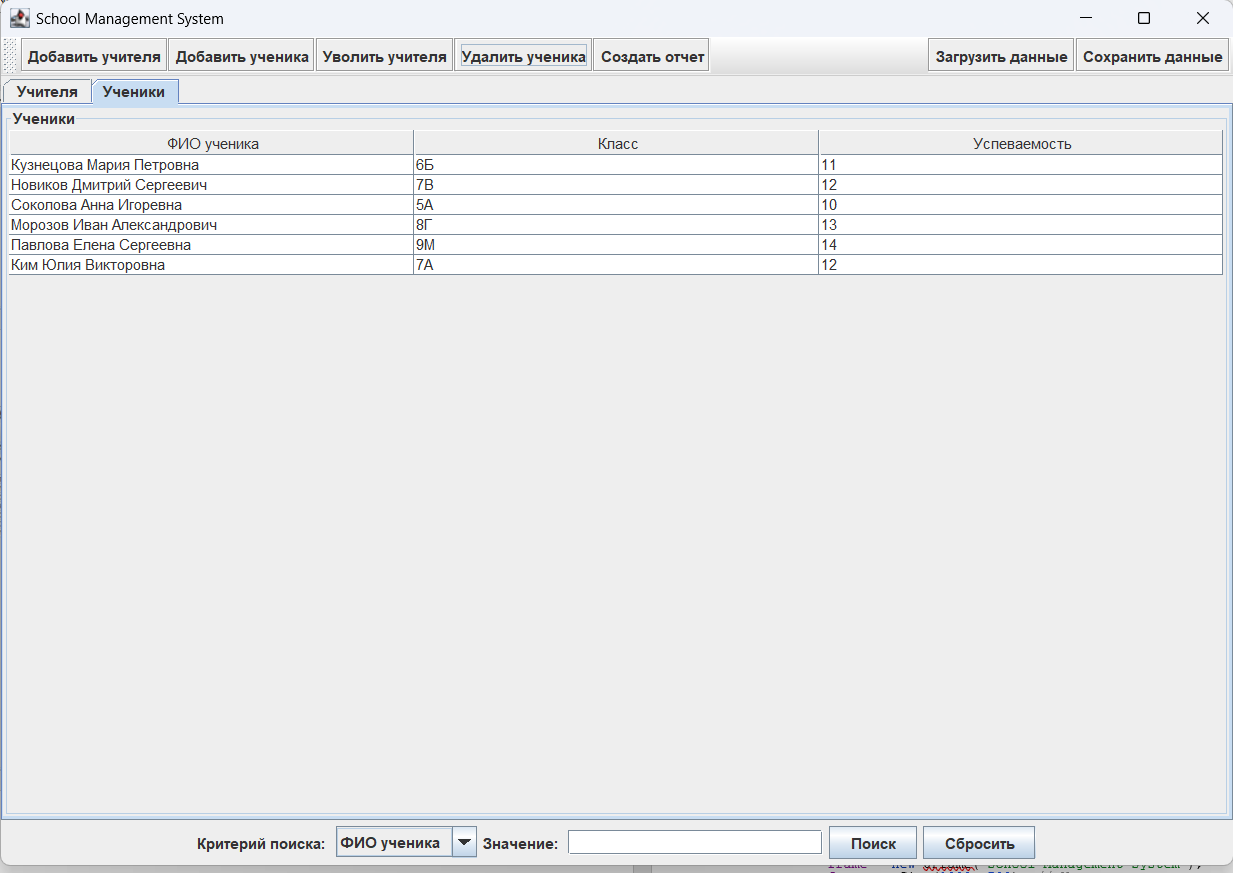


Рисунок 7 – Удалили учеников

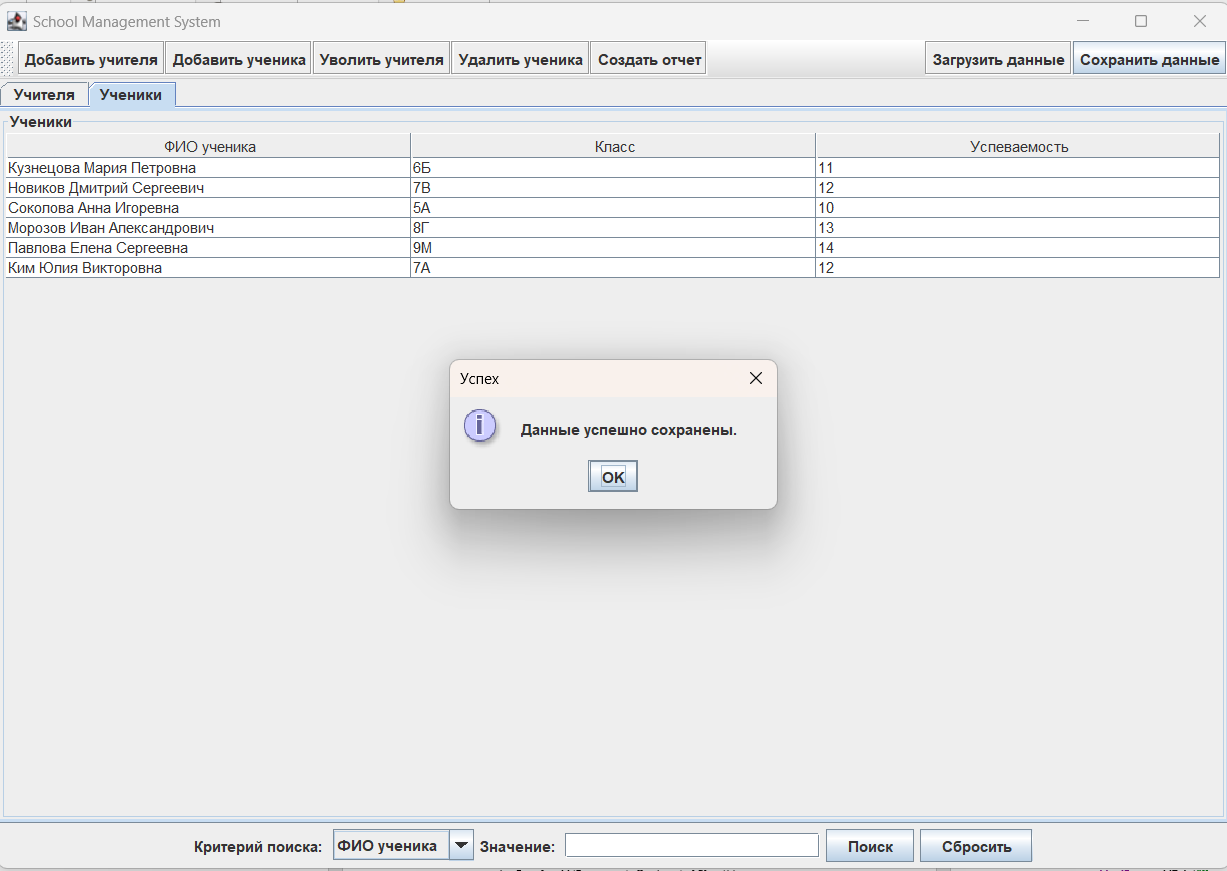


Рисунок 8 – Успешное сохранение в файл *output.txt*

# Текст программы

|  |
| --- |
| import javax.swing.\*; import javax.swing.table.DefaultTableModel; import javax.swing.table.TableRowSorter; import javax.swing.event.ChangeEvent; import javax.swing.event.ChangeListener; import java.awt.\*; import java.awt.event.\*; import java.io.\*; import java.util.ArrayList; import java.util.List;  */\*\*  \* @author Барченков Платон 3312  \* @version 1.1  \*/* public class Main {  private JFrame frame;  private JTable teacherTable, studentTable;  private DefaultTableModel teacherTableModel, studentTableModel;  private JPanel filterPanel;  private JButton addTeacherButton, addStudentButton, deleteTeacherButton, deleteStudentButton, generateReportButton;  private JButton searchButton, resetButton, loadButton, saveButton;  private JComboBox<String> searchCriteria;  private JTextField searchField;  private JScrollPane teacherScrollPane, studentScrollPane;  private JTabbedPane tabbedPane;  private List<String[]> originalTeacherData; *// Исходные данные учителей* private List<String[]> originalStudentData; *// Исходные данные учеников* private TableRowSorter<DefaultTableModel> teacherSorter, studentSorter;   */\*\*  \* Метод для создания и отображения основного окна программы.  \*/* public void SchoolManagementSystem() {  *// Инициализация исходных данных* originalTeacherData = new ArrayList<>();  originalStudentData = new ArrayList<>();   *// Создание главного окна программы* frame = new JFrame("School Management System");  frame.setSize(1000, 700); *// Увеличиваем размер окна для двух таблиц* frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  frame.setLayout(new BorderLayout());   *// Создание панели инструментов с кнопками действий* JToolBar actionPanel = new JToolBar("Toolbar");   *// Кнопки для учителей и учеников* addTeacherButton = new JButton("Добавить учителя");  addStudentButton = new JButton("Добавить ученика");  deleteTeacherButton = new JButton("Уволить учителя");  deleteStudentButton = new JButton("Удалить ученика");  generateReportButton = new JButton("Создать отчет");   *// Кнопки загрузки и сохранения данных* loadButton = new JButton("Загрузить данные");  saveButton = new JButton("Сохранить данные");   *// Добавляем кнопки на панель инструментов слева* actionPanel.add(addTeacherButton);  actionPanel.add(addStudentButton);  actionPanel.add(deleteTeacherButton);  actionPanel.add(deleteStudentButton);  actionPanel.add(generateReportButton);   *// Добавляем гибкое пространство, чтобы следующие кнопки были справа* actionPanel.add(Box.*createHorizontalGlue*());   *// Добавляем кнопки загрузки и сохранения данных справа* actionPanel.add(loadButton);  actionPanel.add(saveButton);   frame.add(actionPanel, BorderLayout.*NORTH*);   *// Определяем столбцы таблицы учителей* String[] teacherColumns = {"ФИО учителя", "Предмет", "Классы"};  *// Исходные данные для таблицы учителей* String[][] initialTeachers = {  {"Иванов Иван Иванович", "Математика", "5А;6Б"},  {"Петрова Анна Сергеевна", "Русский язык", "7В;8Г"},  {"Сидоров Петр Петрович", "История", "9А;10Б"}  };  for (String[] teacher : initialTeachers) {  originalTeacherData.add(teacher);  }   *// Инициализация модели таблицы учителей* teacherTableModel = new DefaultTableModel(teacherColumns, 0);  for (String[] teacher : originalTeacherData) {  teacherTableModel.addRow(teacher);  }  teacherTable = new JTable(teacherTableModel);  teacherScrollPane = new JScrollPane(teacherTable);  teacherScrollPane.setBorder(BorderFactory.*createTitledBorder*("Учителя"));   *// Создание сортировщика для таблицы учителей* teacherSorter = new TableRowSorter<>(teacherTableModel);  teacherTable.setRowSorter(teacherSorter);   *// Определяем столбцы таблицы учеников* String[] studentColumns = {"ФИО ученика", "Класс", "Успеваемость"};  *// Исходные данные для таблицы учеников* String[][] initialStudents = {  {"Смирнов Алексей Иванович", "5А", "Отлично"},  {"Кузнецова Мария Петровна", "6Б", "Хорошо"},  {"Новиков Дмитрий Сергеевич", "7В", "Удовлетворительно"}  };  for (String[] student : initialStudents) {  originalStudentData.add(student);  }   *// Инициализация модели таблицы учеников* studentTableModel = new DefaultTableModel(studentColumns, 0);  for (String[] student : originalStudentData) {  studentTableModel.addRow(student);  }  studentTable = new JTable(studentTableModel);  studentScrollPane = new JScrollPane(studentTable);  studentScrollPane.setBorder(BorderFactory.*createTitledBorder*("Ученики"));   *// Создание сортировщика для таблицы учеников* studentSorter = new TableRowSorter<>(studentTableModel);  studentTable.setRowSorter(studentSorter);   *// Создание вкладок для таблиц* tabbedPane = new JTabbedPane();  tabbedPane.addTab("Учителя", teacherScrollPane);  tabbedPane.addTab("Ученики", studentScrollPane);  frame.add(tabbedPane, BorderLayout.*CENTER*);   *// Создание компонентов для панели поиска и фильтрации данных* searchCriteria = new JComboBox<>(new String[]{  "ФИО учителя", "Предмет", "Классы",  "ФИО ученика", "Класс ученика", "Успеваемость"  });  searchField = new JTextField(20);  searchButton = new JButton("Поиск");  resetButton = new JButton("Сбросить");   *// Панель фильтрации* filterPanel = new JPanel();  filterPanel.add(new JLabel("Критерий поиска: "));  filterPanel.add(searchCriteria);  filterPanel.add(new JLabel("Значение: "));  filterPanel.add(searchField);  filterPanel.add(searchButton);  filterPanel.add(resetButton);  frame.add(filterPanel, BorderLayout.*SOUTH*);   *// Действие при переключении вкладок для обновления критериев поиска* tabbedPane.addChangeListener(new ChangeListener() {  @Override  public void stateChanged(ChangeEvent e) {  updateSearchCriteria();  }  });   *// Инициализация критериев поиска по текущей вкладке* updateSearchCriteria();   *// Действие при нажатии кнопки "Поиск"* searchButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  String criterion = (String) searchCriteria.getSelectedItem();  String value = searchField.getText().trim();  searchTable(criterion, value);  }  });   *// Действие при нажатии кнопки "Сбросить"* resetButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  resetTable();  }  });   *// Действие при нажатии кнопки "Добавить учителя"* addTeacherButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  String teacherName = JOptionPane.*showInputDialog*(frame, "Введите ФИО учителя:");  if (teacherName == null || teacherName.trim().isEmpty()) {  JOptionPane.*showMessageDialog*(frame, "ФИО учителя не может быть пустым.", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  return;  }   String subject = JOptionPane.*showInputDialog*(frame, "Введите предмет:");  if (subject == null || subject.trim().isEmpty()) {  JOptionPane.*showMessageDialog*(frame, "Предмет не может быть пустым.", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  return;  }   String classes = JOptionPane.*showInputDialog*(frame, "Введите классы (разделенные точкой с запятой ';'):");  if (classes == null || classes.trim().isEmpty()) {  JOptionPane.*showMessageDialog*(frame, "Классы не могут быть пустыми.", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  return;  }   String[] newTeacher = {teacherName.trim(), subject.trim(), classes.trim()};  teacherTableModel.addRow(newTeacher);  originalTeacherData.add(newTeacher);  }  });   *// Действие при нажатии кнопки "Удалить учителя"* deleteTeacherButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  int selectedRow = teacherTable.getSelectedRow();  if (selectedRow != -1) {  *// Преобразование индекса с учёта сортировки* selectedRow = teacherTable.convertRowIndexToModel(selectedRow);  teacherTableModel.removeRow(selectedRow);  originalTeacherData.remove(selectedRow);  } else {  JOptionPane.*showMessageDialog*(frame, "Пожалуйста, выберите учителя для удаления", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  }  }  });   *// Действие при нажатии кнопки "Добавить ученика"* addStudentButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  String studentName = JOptionPane.*showInputDialog*(frame, "Введите ФИО ученика:");  if (studentName == null || studentName.trim().isEmpty()) {  JOptionPane.*showMessageDialog*(frame, "ФИО ученика не может быть пустым.", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  return;  }   String studentClass = JOptionPane.*showInputDialog*(frame, "Введите класс:");  if (studentClass == null || studentClass.trim().isEmpty()) {  JOptionPane.*showMessageDialog*(frame, "Класс не может быть пустым.", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  return;  }   String performance = JOptionPane.*showInputDialog*(frame, "Введите успеваемость:");  if (performance == null || performance.trim().isEmpty()) {  JOptionPane.*showMessageDialog*(frame, "Успеваемость не может быть пустой.", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  return;  }   String[] newStudent = {studentName.trim(), studentClass.trim(), performance.trim()};  studentTableModel.addRow(newStudent);  originalStudentData.add(newStudent);  }  });   *// Действие при нажатии кнопки "Удалить ученика"* deleteStudentButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  int selectedRow = studentTable.getSelectedRow();  if (selectedRow != -1) {  *// Преобразование индекса с учёта сортировки* selectedRow = studentTable.convertRowIndexToModel(selectedRow);  studentTableModel.removeRow(selectedRow);  originalStudentData.remove(selectedRow);  } else {  JOptionPane.*showMessageDialog*(frame, "Пожалуйста, выберите ученика для удаления", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  }  }  });   *// Действие при нажатии кнопки "Загрузить данные"* loadButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  loadDataFromFile();  }  });   *// Действие при нажатии кнопки "Сохранить данные"* saveButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  saveDataToFile();  }  });   *// Делаем главное окно видимым* frame.setVisible(true);  }   */\*\*  \* Обновляет критерии поиска в зависимости от выбранной вкладки.  \*/* private void updateSearchCriteria() {  int selectedIndex = tabbedPane.getSelectedIndex();  searchCriteria.removeAllItems();   if (selectedIndex == 0) { *// Учителя* searchCriteria.addItem("ФИО учителя");  searchCriteria.addItem("Предмет");  searchCriteria.addItem("Классы");  } else if (selectedIndex == 1) { *// Ученики* searchCriteria.addItem("ФИО ученика");  searchCriteria.addItem("Класс ученика");  searchCriteria.addItem("Успеваемость");  }  }   */\*\*  \* Метод для фильтрации данных в таблице на основе критерия и значения поиска.  \*  \* @param criterion Критерий поиска.  \* @param value Значение для поиска.  \*/* private void searchTable(String criterion, String value) {  if (value.isEmpty()) {  JOptionPane.*showMessageDialog*(frame, "Поле поиска не может быть пустым.", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  return;  }   int selectedIndex = tabbedPane.getSelectedIndex();   if (selectedIndex == 0) { *// Учителя* int columnIndex = -1;  switch (criterion) {  case "ФИО учителя":  columnIndex = 0;  break;  case "Предмет":  columnIndex = 1;  break;  case "Классы":  columnIndex = 2;  break;  }   if (columnIndex != -1) {  teacherSorter.setRowFilter(RowFilter.*regexFilter*("(?i)" + value, columnIndex));  }  } else if (selectedIndex == 1) { *// Ученики* int columnIndex = -1;  switch (criterion) {  case "ФИО ученика":  columnIndex = 0;  break;  case "Класс ученика":  columnIndex = 1;  break;  case "Успеваемость":  columnIndex = 2;  break;  }   if (columnIndex != -1) {  studentSorter.setRowFilter(RowFilter.*regexFilter*("(?i)" + value, columnIndex));  }  }  }   */\*\*  \* Метод для сброса фильтров и восстановления исходных данных.  \*/* private void resetTable() {  *// Сброс фильтра для учителей* teacherSorter.setRowFilter(null);  *// Сброс фильтра для учеников* studentSorter.setRowFilter(null);  *// Очистка поля поиска* searchField.setText("");  }   */\*\*  \* Метод для загрузки данных из файла, выбранного пользователем.  \*/* private void loadDataFromFile() {  JFileChooser fileChooser = new JFileChooser();  fileChooser.setDialogTitle("Выберите текстовый файл для загрузки данных");  int userSelection = fileChooser.showOpenDialog(frame);   if (userSelection == JFileChooser.*APPROVE\_OPTION*) {  File fileToLoad = fileChooser.getSelectedFile();  try (BufferedReader br = new BufferedReader(new FileReader(fileToLoad))) {  String line;  boolean isTeacherSection = false;  boolean isStudentSection = false;   List<String[]> loadedTeachers = new ArrayList<>();  List<String[]> loadedStudents = new ArrayList<>();   while ((line = br.readLine()) != null) {  line = line.trim();  if (line.isEmpty()) continue;   if (line.equalsIgnoreCase("# Teachers")) {  isTeacherSection = true;  isStudentSection = false;  br.readLine(); *// Пропускаем заголовок столбцов* continue;  } else if (line.equalsIgnoreCase("# Students")) {  isTeacherSection = false;  isStudentSection = true;  br.readLine(); *// Пропускаем заголовок столбцов* continue;  }   if (isTeacherSection) {  String[] parts = line.split(",", 3);  if (parts.length == 3) {  loadedTeachers.add(new String[]{parts[0].trim(), parts[1].trim(), parts[2].trim()});  }  } else if (isStudentSection) {  String[] parts = line.split(",", 3);  if (parts.length == 3) {  loadedStudents.add(new String[]{parts[0].trim(), parts[1].trim(), parts[2].trim()});  }  }  }   *// Обновляем таблицы учителей* teacherTableModel.setRowCount(0);  originalTeacherData.clear();  for (String[] teacher : loadedTeachers) {  teacherTableModel.addRow(teacher);  originalTeacherData.add(teacher);  }   *// Обновляем таблицы учеников* studentTableModel.setRowCount(0);  originalStudentData.clear();  for (String[] student : loadedStudents) {  studentTableModel.addRow(student);  originalStudentData.add(student);  }   JOptionPane.*showMessageDialog*(frame, "Данные успешно загружены.", "Успех", JOptionPane.*INFORMATION\_MESSAGE*);   } catch (IOException ex) {  JOptionPane.*showMessageDialog*(frame, "Ошибка при загрузке данных: " + ex.getMessage(), "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  }  }  }   */\*\*  \* Метод для сохранения данных в файл, выбранный пользователем.  \*/* private void saveDataToFile() {  JFileChooser fileChooser = new JFileChooser();  fileChooser.setDialogTitle("Сохраните данные в текстовый файл");  int userSelection = fileChooser.showSaveDialog(frame);   if (userSelection == JFileChooser.*APPROVE\_OPTION*) {  File fileToSave = fileChooser.getSelectedFile();  try (BufferedWriter bw = new BufferedWriter(new FileWriter(fileToSave))) {  *// Записываем учителей* bw.write("# Teachers");  bw.newLine();  bw.write("ФИО учителя,Предмет,Классы");  bw.newLine();  for (String[] teacher : originalTeacherData) {  String line = String.*join*(",", teacher);  bw.write(line);  bw.newLine();  }   bw.newLine(); *// Пустая строка между разделами   // Записываем учеников* bw.write("# Students");  bw.newLine();  bw.write("ФИО ученика,Класс,Успеваемость");  bw.newLine();  for (String[] student : originalStudentData) {  String line = String.*join*(",", student);  bw.write(line);  bw.newLine();  }   JOptionPane.*showMessageDialog*(frame, "Данные успешно сохранены.", "Успех", JOptionPane.*INFORMATION\_MESSAGE*);   } catch (IOException ex) {  JOptionPane.*showMessageDialog*(frame, "Ошибка при сохранении данных: " + ex.getMessage(), "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  }  }  }   */\*\*  \* Точка входа в программу. Запуск приложения.  \*  \* @param args Аргументы командной строки (не используются).  \*/* public static void main(String[] args) {  *// Запуск интерфейса в потоке обработки событий Swing* SwingUtilities.*invokeLater*(new Runnable() {  public void run() {  new Main().SchoolManagementSystem();  }  });  } } |

# Приложение

Репозиторий: https://github.com/PlatonBarchenkov/OOP\_lab\_05.git