**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 № 3**

**«Обработка событий»**

**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](#_Toc180317358)

[Описание действий слушателей 3](#_Toc180317359)

[Пример работы слушателей 4](#_Toc180317360)

[Текст программы 8](#_Toc180317361)

[Приложение 11](#_Toc180317362)

# Цель работы

Знакомство со способами подключения слушателей событий к графическим компонентам пользовательского интерфейса.

# Описание действий слушателей

1. Слушатель для кнопки "Добавить учителя" (addTeacherButton):

При нажатии на кнопку "Добавить учителя" открывается три последовательных диалоговых окна, где пользователь вводит ФИО учителя, предмет, который он преподает, и классы, в которых он работает. После того как все данные будут введены, новая запись с информацией о преподавателе добавляется в таблицу. В дальнейшем это действие можно расширить для более сложного добавления данных через формы.

1. Слушатель для кнопки "Удалить учителя" (deleteTeacherButton):

При нажатии на кнопку "Удалить учителя" происходит удаление выбранной строки из таблицы, если строка была выбрана. Если никакая строка не была выбрана, отображается диалоговое окно с сообщением об ошибке, информирующее пользователя о необходимости выбрать строку для удаления. В дальнейшем это действие можно улучшить добавлением подтверждения перед удалением.

1. Слушатель для кнопки "Поиск" (searchButton):

При нажатии на кнопку "Поиск" происходит фильтрация данных в таблице на основе выбранного критерия (например, ФИО учителя, предмет или класс) и введенного значения. Таблица обновляется, отображая только те строки, которые соответствуют критериям поиска. В дальнейшем можно добавить более сложные фильтры и расширить функциональность поиска.

1. Слушатель для кнопки "Сбросить" (resetButton):

При нажатии на кнопку "Сбросить" снимается текущая фильтрация, и таблица возвращается к своему исходному состоянию, показывая все данные без применения каких-либо фильтров. Это действие можно дополнить, например, подтверждением сброса фильтров.

# Пример работы слушателей

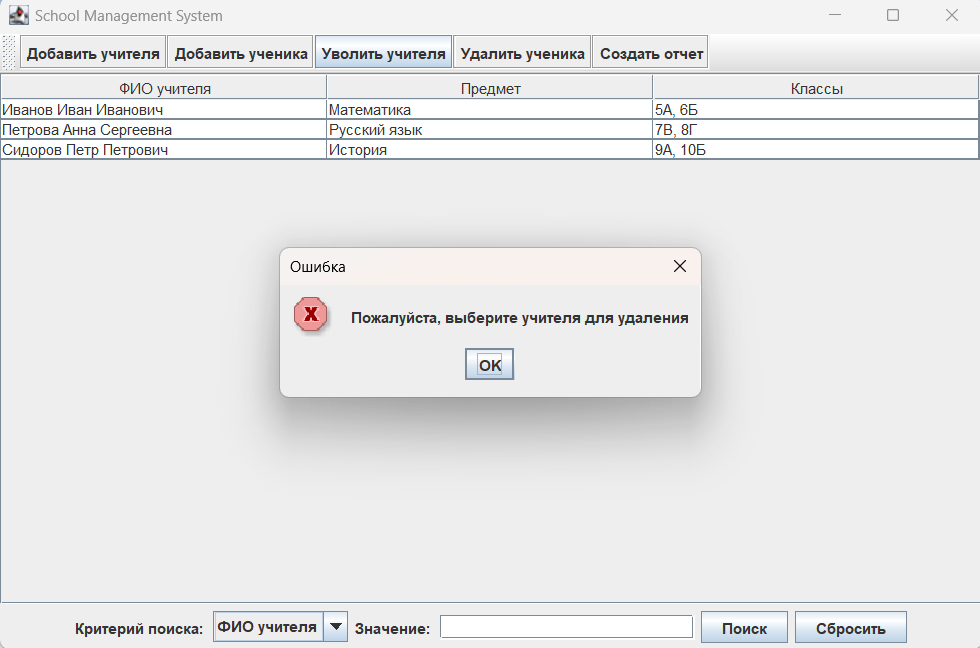


Рисунок 1 – нажатие кнопки уволить при не выбранном учителе

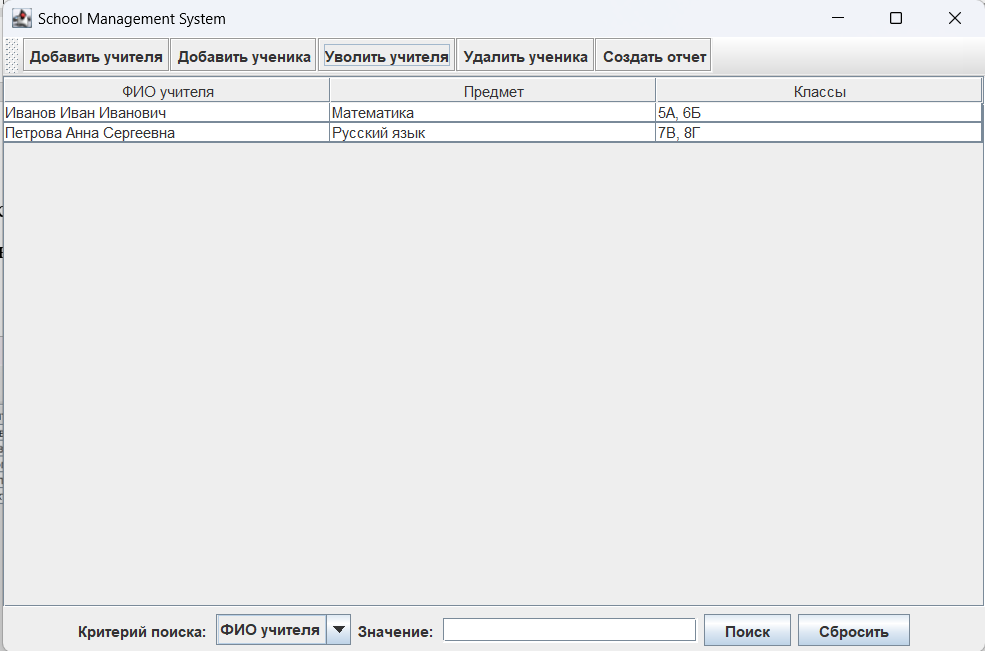


Рисунок 2 – уволили учителя истории

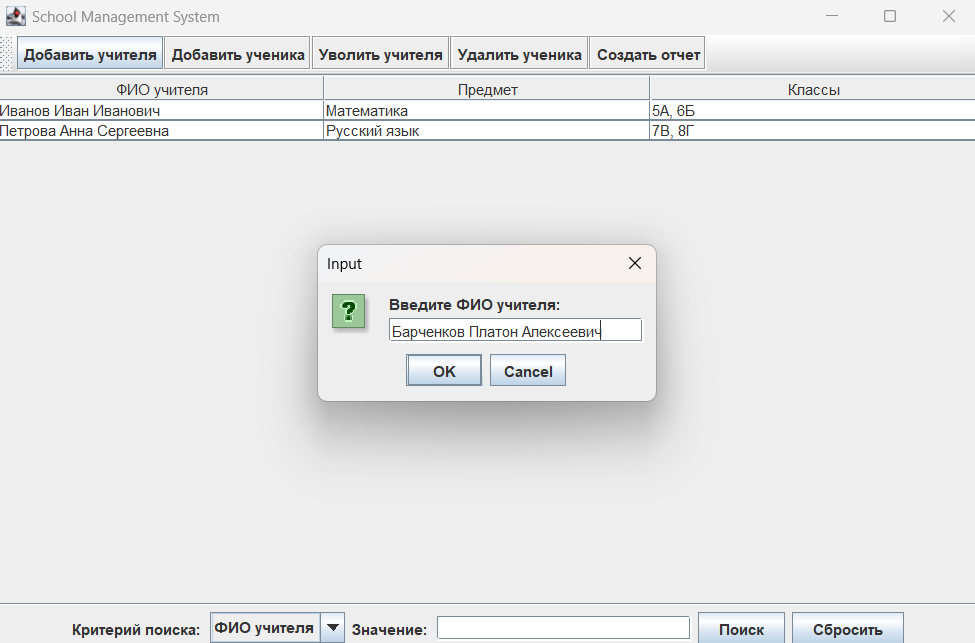


Рисунок 3 – введение ФИО нового учителя

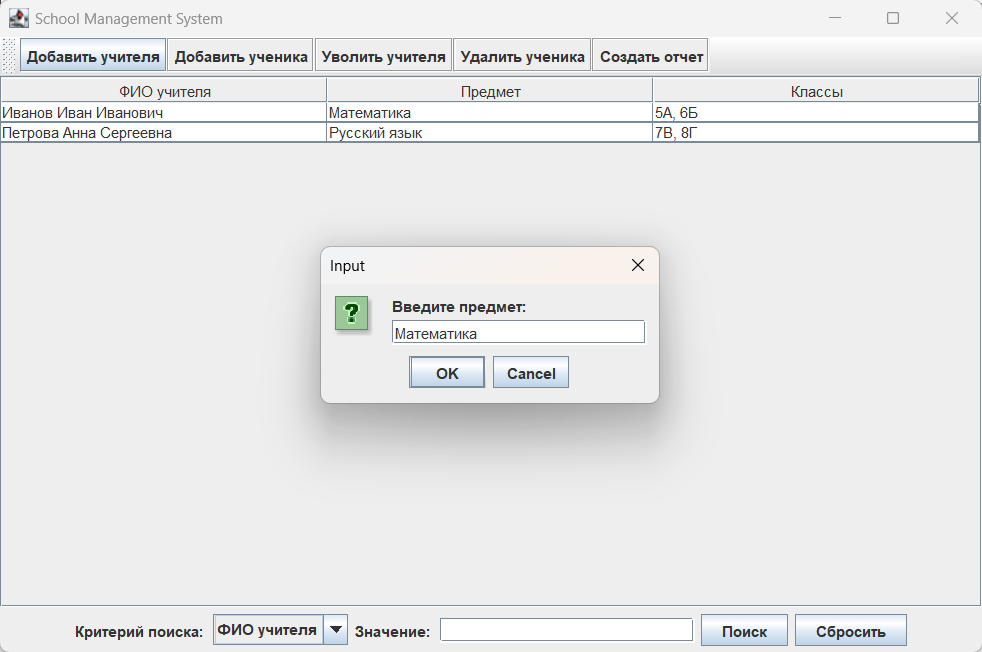


Рисунок 4 – введение предмета, который ведет новый учитель

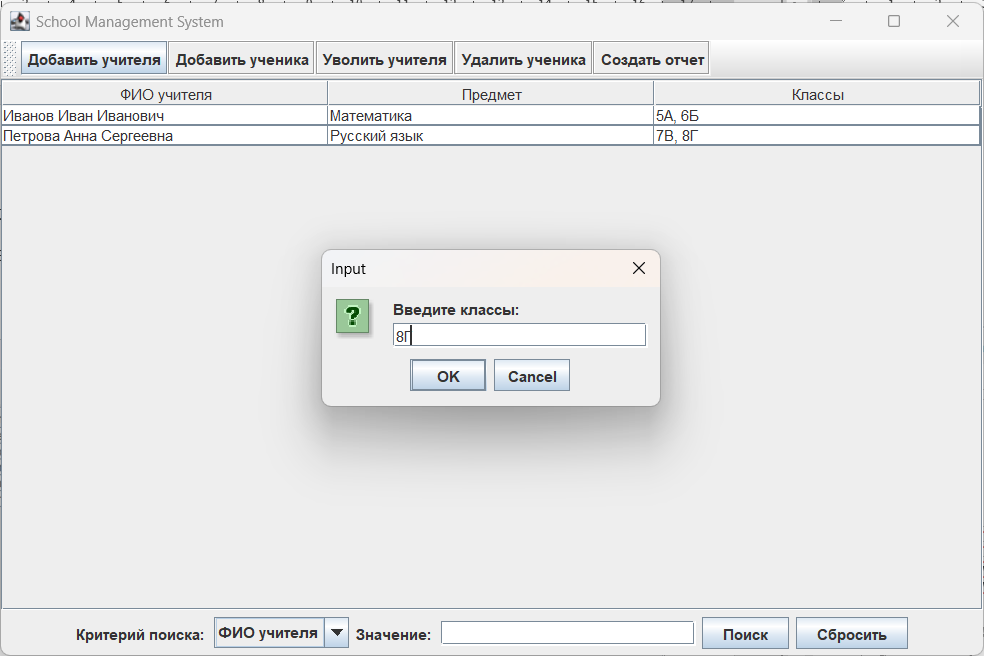


Рисунок 5 – введение класса, в котором ведет новый учитель

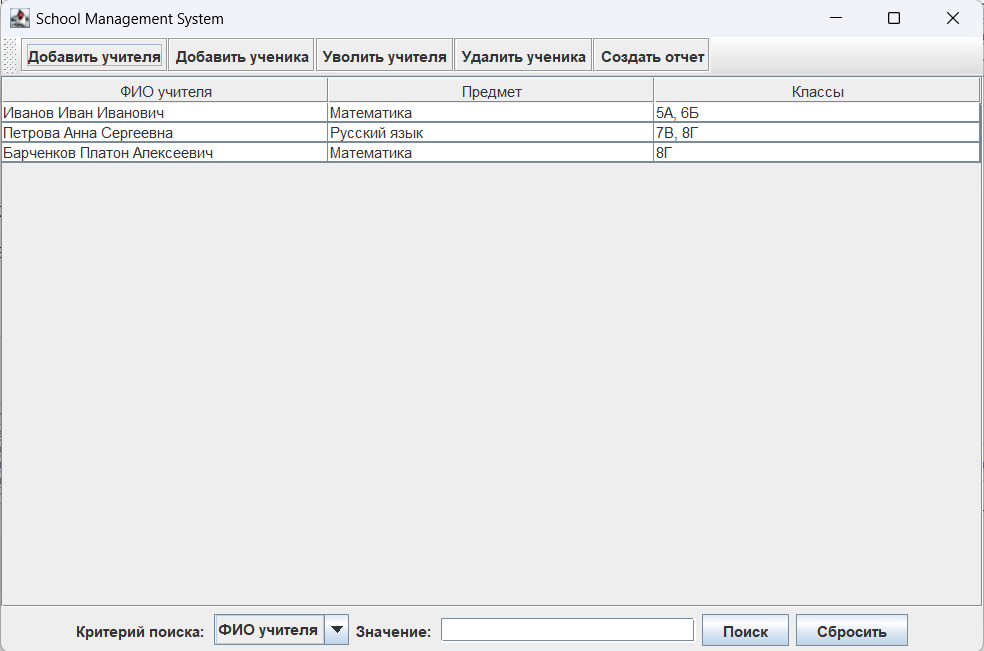


Рисунок 6 – добавлен новый учитель

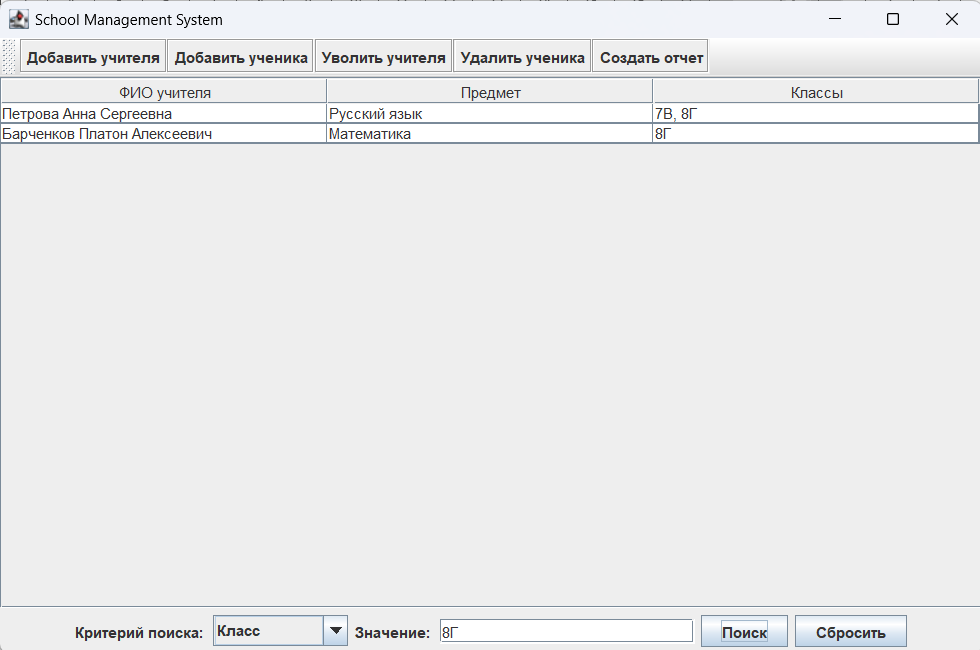


Рисунок 7 – результат поиска по классу

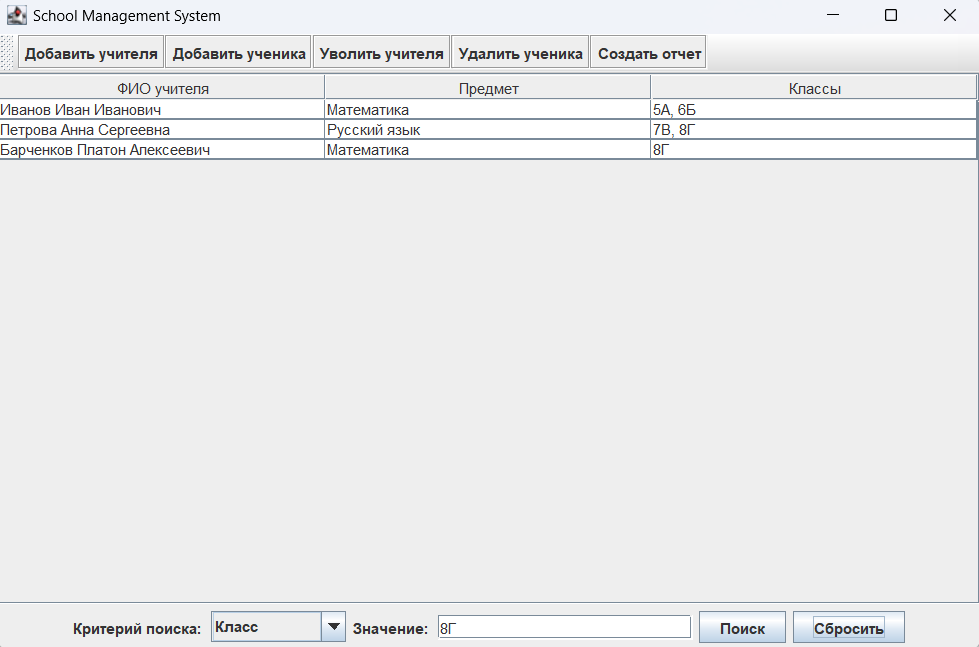


Рисунок 8 – сброс параметров поиска

# Текст программы

|  |
| --- |
| import javax.swing.\*; import javax.swing.table.DefaultTableModel; import javax.swing.table.TableRowSorter; import java.awt.\*; import java.awt.event.\*;  */\*\*  \* Программа для управления данными учителей и учеников в системе управления школой.  \* Содержит функции добавления, удаления учителей, а также поиска и фильтрации данных.  \*  \* @author Барченков Платон 3312  \* @version 1.0  \*/* public class Main {  private JFrame frame;  private JTable teacherTable;  private DefaultTableModel teacherTableModel;  private JPanel filterPanel;  private JButton addTeacherButton, addStudentButton, deleteTeacherButton, deleteStudentButton, generateReportButton;  private JButton searchButton, resetButton;  private JComboBox<String> searchCriteria;  private JTextField searchField;  private JScrollPane teacherScrollPane;  private Object[][] originalTeacherData; *// Массив для хранения исходных данных таблицы учителей   /\*\*  \* Метод для создания и отображения основного окна программы.  \* Создает таблицу с данными учителей и панель инструментов с кнопками для управления данными.  \*/* public void SchoolManagementSystem() {  *// Создание главного окна программы* frame = new JFrame("School Management System");  frame.setSize(800, 600); *// Устанавливаем размер окна* frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*); *// Закрытие окна завершает программу* frame.setLayout(new BorderLayout()); *// Устанавливаем BorderLayout для главного окна   // Создание панели инструментов с кнопками действий* JToolBar actionPanel = new JToolBar("Toolbar");  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");   *// Добавляем кнопки на панель инструментов* actionPanel.add(addTeacherButton);  actionPanel.add(addStudentButton);  actionPanel.add(deleteTeacherButton);  actionPanel.add(deleteStudentButton);  actionPanel.add(generateReportButton);   frame.add(actionPanel, BorderLayout.*NORTH*); *// Размещаем панель инструментов сверху   // Определяем столбцы таблицы* String[] teacherColumns = {"ФИО учителя", "Предмет", "Классы"};  *// Исходные данные для таблицы учителей* originalTeacherData = new Object[][]{  {"Иванов Иван Иванович", "Математика", "5А, 6Б"},  {"Петрова Анна Сергеевна", "Русский язык", "7В, 8Г"},  {"Сидоров Петр Петрович", "История", "9А, 10Б"}  };   *// Инициализация модели таблицы с исходными данными* teacherTableModel = new DefaultTableModel(originalTeacherData, teacherColumns);  teacherTable = new JTable(teacherTableModel);  teacherScrollPane = new JScrollPane(teacherTable);  frame.add(teacherScrollPane, BorderLayout.*CENTER*);   *// Создание компонентов для панели поиска и фильтрации данных* searchCriteria = new JComboBox<>(new String[]{"ФИО учителя", "Предмет", "Класс"});  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 нажатии кнопки "Поиск"* searchButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *// Получаем выбранный критерий и введенное значение для поиска* String criterion = (String) searchCriteria.getSelectedItem();  String value = searchField.getText().trim();  *// Выполняем фильтрацию таблицы на основе выбранного критерия* searchTable(criterion, value);  }  });   *// Действие при нажатии кнопки "Сбросить"* resetButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *// Сбрасываем фильтрацию и возвращаем исходные данные* resetTable();  }  });   *// Действие при нажатии кнопки "Добавить учителя"* addTeacherButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  *// Диалоговые окна для ввода данных о новом учителе* String teacherName = JOptionPane.*showInputDialog*(frame, "Введите ФИО учителя:");  String subject = JOptionPane.*showInputDialog*(frame, "Введите предмет:");  String classes = JOptionPane.*showInputDialog*(frame, "Введите классы:");   *// Проверка, что введены все данные* if (teacherName != null && subject != null && classes != null) {  *// Добавляем новую запись в таблицу учителей* teacherTableModel.addRow(new Object[]{teacherName, subject, classes});  *// Обновляем массив с оригинальными данными* addOriginalTeacherData(new Object[]{teacherName, subject, classes});  }  }  });   *// Действие при нажатии кнопки "Удалить учителя"* deleteTeacherButton.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  int selectedRow = teacherTable.getSelectedRow(); *// Получаем индекс выбранной строки  // Проверяем, что строка была выбрана* if (selectedRow != -1) {  *// Удаляем выбранную строку из таблицы* teacherTableModel.removeRow(selectedRow);  *// Обновляем исходные данные* removeOriginalTeacherData(selectedRow);  } else {  *// Показываем сообщение, если не выбрана строка* JOptionPane.*showMessageDialog*(frame, "Пожалуйста, выберите учителя для удаления", "Ошибка", JOptionPane.*ERROR\_MESSAGE*);  }  }  });   *// Делаем главное окно видимым* frame.setVisible(true);  }   */\*\*  \* Метод для фильтрации данных в таблице на основе критерия и значения поиска.  \*  \* @param criterion Критерий поиска (например, "ФИО учителя").  \* @param value Значение, по которому нужно фильтровать.  \*/* private void searchTable(String criterion, String value) {  if (!value.isEmpty()) { *// Проверяем, что поле поиска не пустое* DefaultTableModel model = (DefaultTableModel) teacherTable.getModel();  TableRowSorter<DefaultTableModel> sorter = new TableRowSorter<>(model); *// Сортировщик для фильтрации* teacherTable.setRowSorter(sorter); *// Применяем сортировщик к таблице   // Определяем индекс столбца для поиска в зависимости от критерия* int columnIndex = -1;  switch (criterion) {  case "ФИО учителя":  columnIndex = 0;  break;  case "Предмет":  columnIndex = 1;  break;  case "Класс":  columnIndex = 2;  break;  }   *// Применяем фильтрацию по значению* if (columnIndex != -1) {  sorter.setRowFilter(RowFilter.*regexFilter*("(?i)" + value, columnIndex));  }  }  }   */\*\*  \* Данный метод удаляет текущие фильтры, очищает все строки таблицы,  \* а затем восстанавливает в нее все исходные данные о преподавателях,  \* которые были до применения фильтров.  \*/* private void resetTable() {  teacherTable.setRowSorter(null); *// Убираем фильтр из таблицы* teacherTableModel.setRowCount(0); *// Очищаем таблицу  // Восстанавливаем все исходные данные* for (Object[] row : originalTeacherData) {  teacherTableModel.addRow(row);  }  }   */\*\*  \* Добавляет нового учителя в массив с исходными данными.  \*  \* @param newRow Массив данных о новом учителе.  \*/* private void addOriginalTeacherData(Object[] newRow) {  Object[][] newOriginalData = new Object[originalTeacherData.length + 1][3];  System.*arraycopy*(originalTeacherData, 0, newOriginalData, 0, originalTeacherData.length);  newOriginalData[originalTeacherData.length] = newRow;  originalTeacherData = newOriginalData; *// Обновляем массив оригинальных данных* }   */\*\*  \* Удаляет учителя из массива с исходными данными.  \*  \* @param rowIndex Индекс строки, которая должна быть удалена.  \*/* private void removeOriginalTeacherData(int rowIndex) {  Object[][] newOriginalData = new Object[originalTeacherData.length - 1][3];  for (int i = 0, j = 0; i < originalTeacherData.length; i++) {  if (i != rowIndex) {  newOriginalData[j++] = originalTeacherData[i];  }  }  originalTeacherData = newOriginalData; *// Обновляем массив оригинальных данных* }   */\*\*  \* Точка входа в программу. Запуск приложения.  \*  \* @param args Аргументы командной строки (не используются).  \*/* public static void main(String[] args) {  new Main().SchoolManagementSystem();  } } |

# Приложение

Видео: https://rutube.ru/video/private/f6ebf500892ba901994be49bcd3e6d26/?p=XMzUwf-J3a0bFRusOqNTIg

Репозиторий: https://github.com/PlatonBarchenkov/OOP\_lab\_03.git