**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: Шарапов И.Д.

Факультет: КТИ

Группа: №3312

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

Санкт-Петербург

2024

**Содержание**

[Цель работы 3](#_Toc180179489)

[Описание действий слушателей 3](#_Toc180179490)

[Пример работы слушателей 4](#_Toc180179491)

[Текст программы 7](#_Toc180179492)

[Приложение 11](#_Toc180179493)

# Цель работы

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

# Описание действий слушателей

1. Слушатель для кнопки "Добавить" (*addDriverButton*):

При нажатии на кнопку "Добавить" открывается диалоговое окно с информационным сообщением. Это действие показывает пользователю, что в будущем при нажатии на эту кнопку будет реализована возможность добавления новой записи о водителе и его нарушении в таблицу.

1. Слушатель для кнопки "Редактировать" (*editDriverButton*):

При нажатии на кнопку "Редактировать" открывается диалоговое окно с сообщением о редактировании. В дальнейшем эта кнопка может быть расширена для изменения выбранной записи в таблице данных.

1. Слушатель для кнопки "Удалить" (*deleteDriverButton*):

При нажатии на кнопку "Удалить" также открывается диалоговое окно с информацией об удалении записи. Это действие можно расширить для фактического удаления выбранной записи из таблицы.

1. Слушатель для кнопки "Загрузить" (*loadDriverButton*):

Слушатель для этой кнопки реализует открытие диалогового окна выбора файла с помощью *JFileChooser*. Пользователь может выбрать файл, содержащий данные, и после выбора данные из файла загружаются в таблицу программы. В случае успеха выводится сообщение об успешной загрузке, в случае ошибки – уведомление о проблеме с файлом.

1. Слушатель для кнопки "Сохранить" (*saveDriverButton*):

При нажатии на кнопку "Сохранить" открывается диалоговое окно для выбора места сохранения файла. Программа сохраняет данные из таблицы в текстовый файл с разделением значений через табуляцию. В случае успеха выводится сообщение об успешном сохранении, а при возникновении ошибки – уведомление о проблеме.

# Пример работы слушателей

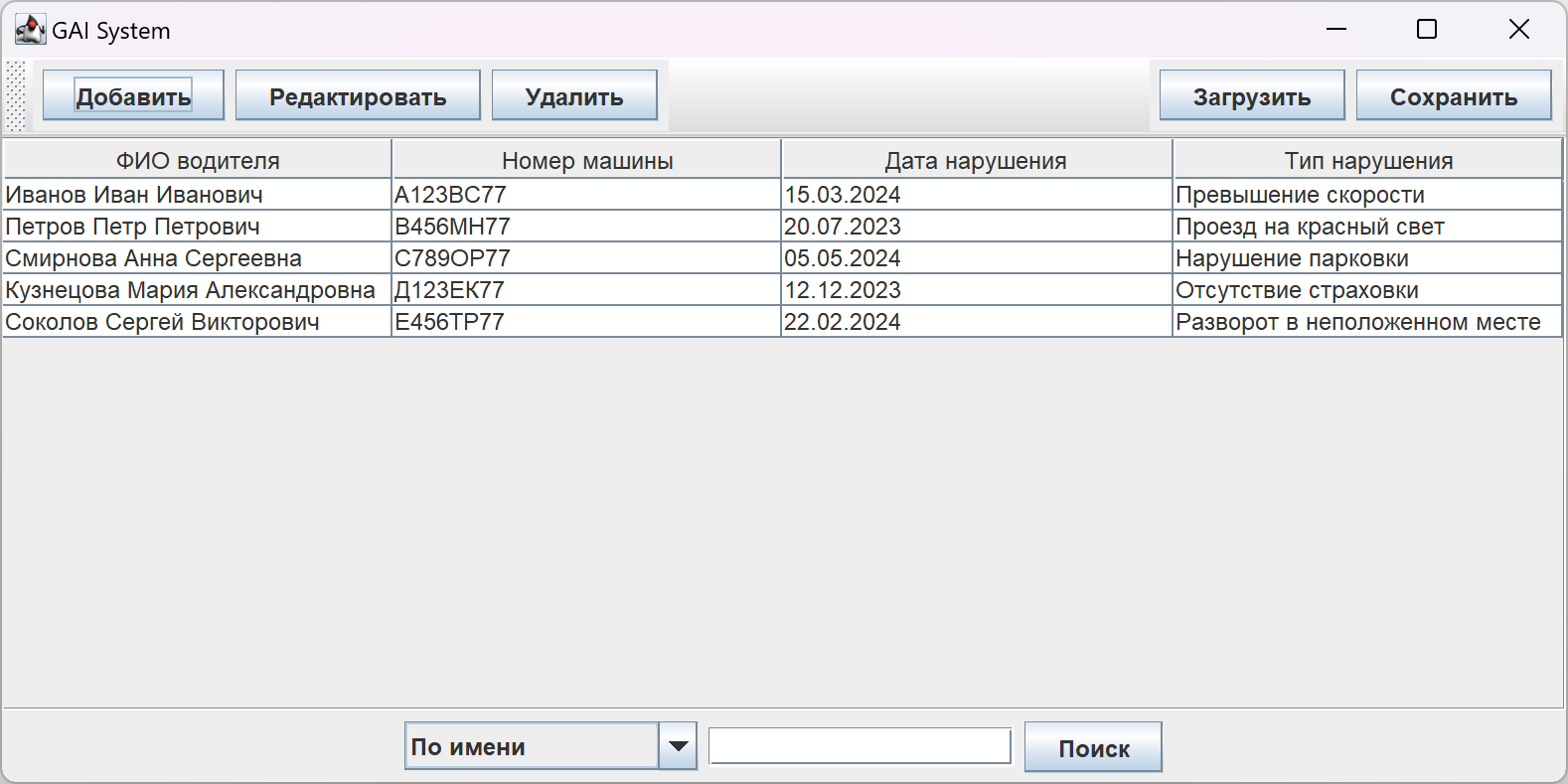


Рисунок 1 – Изменённое расположение кнопок

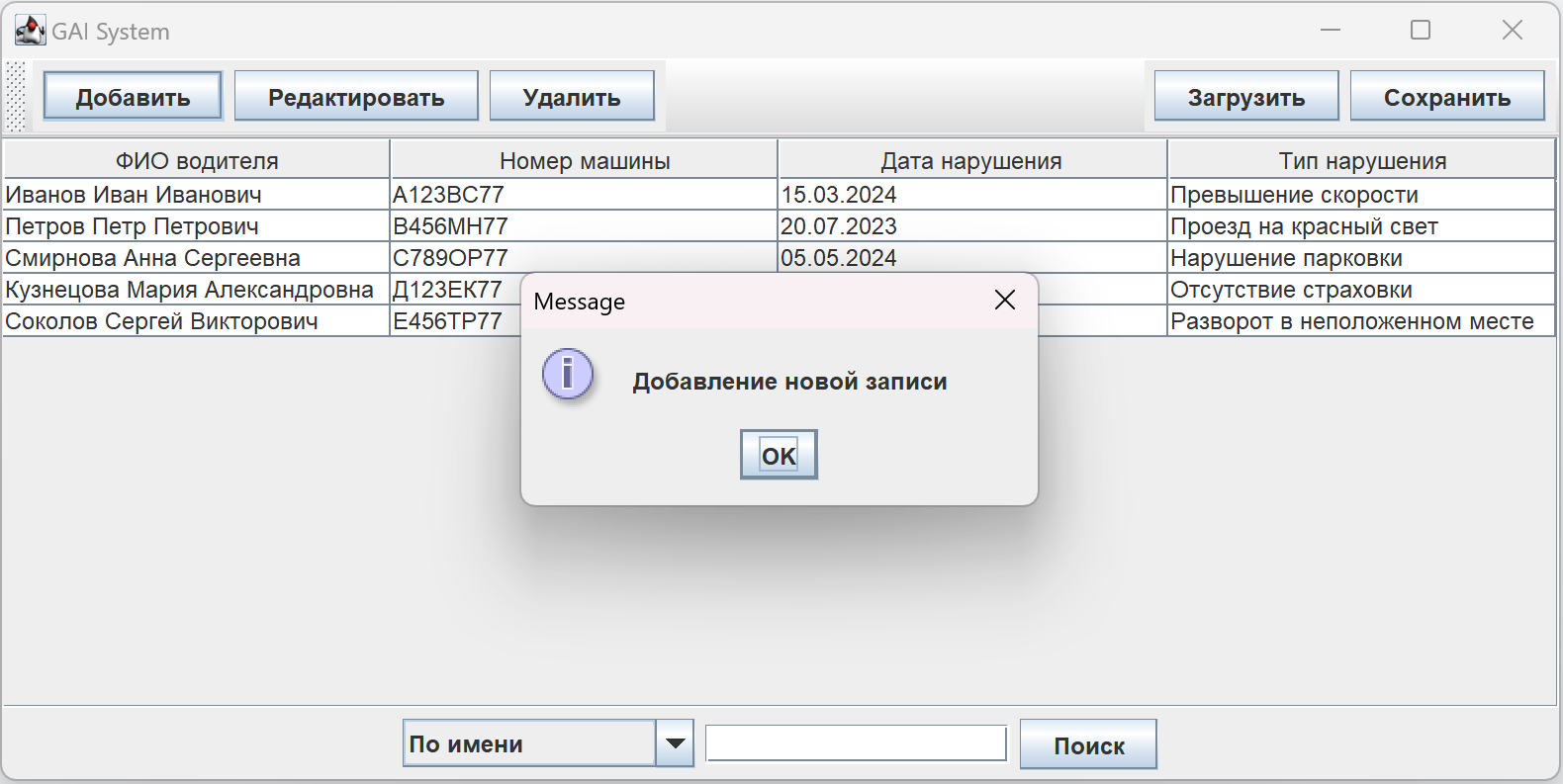


Рисунок 2 – Работа кнопки «Добавить»

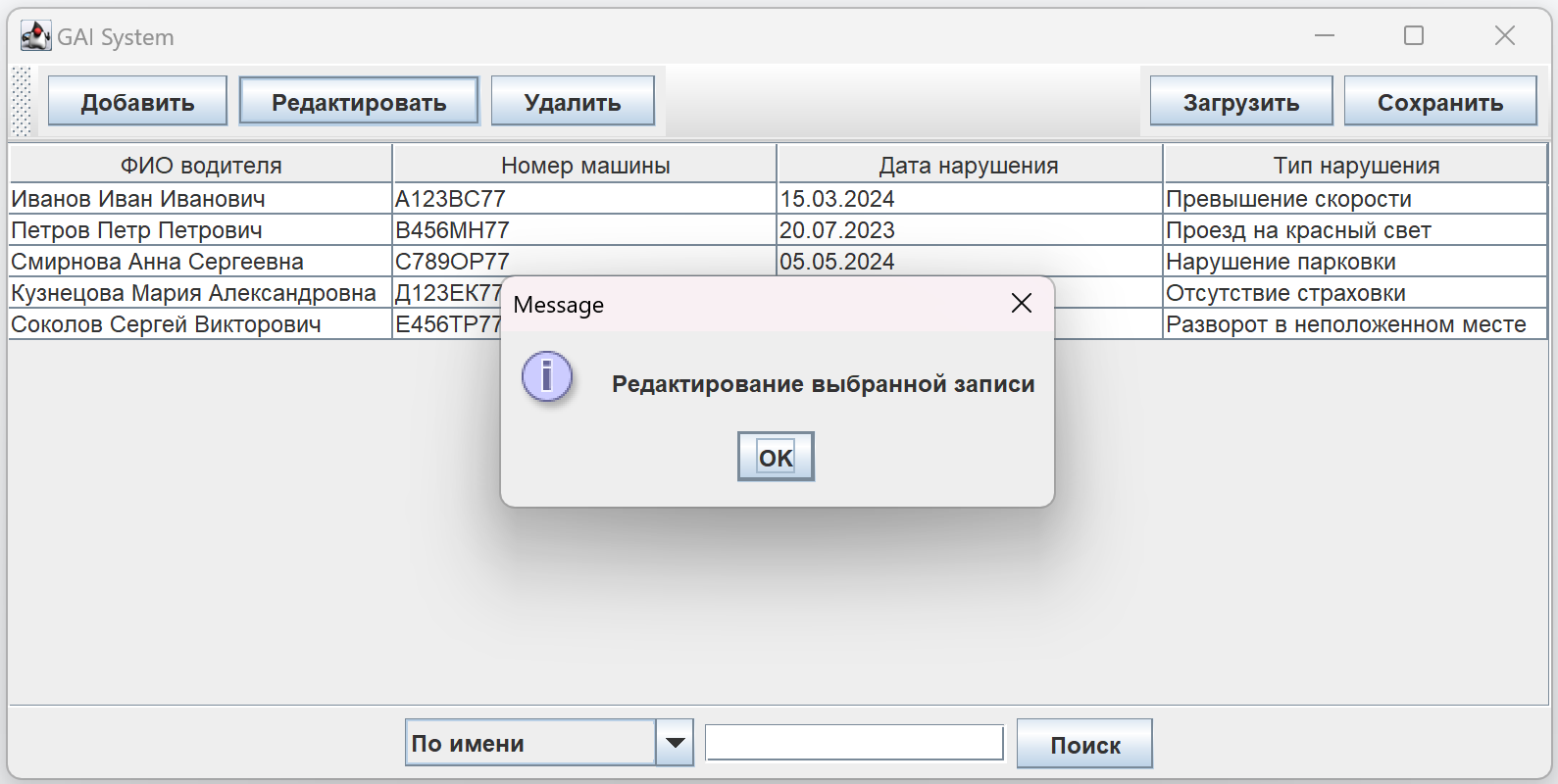


Рисунок 3 – Работа кнопки «Редактировать»

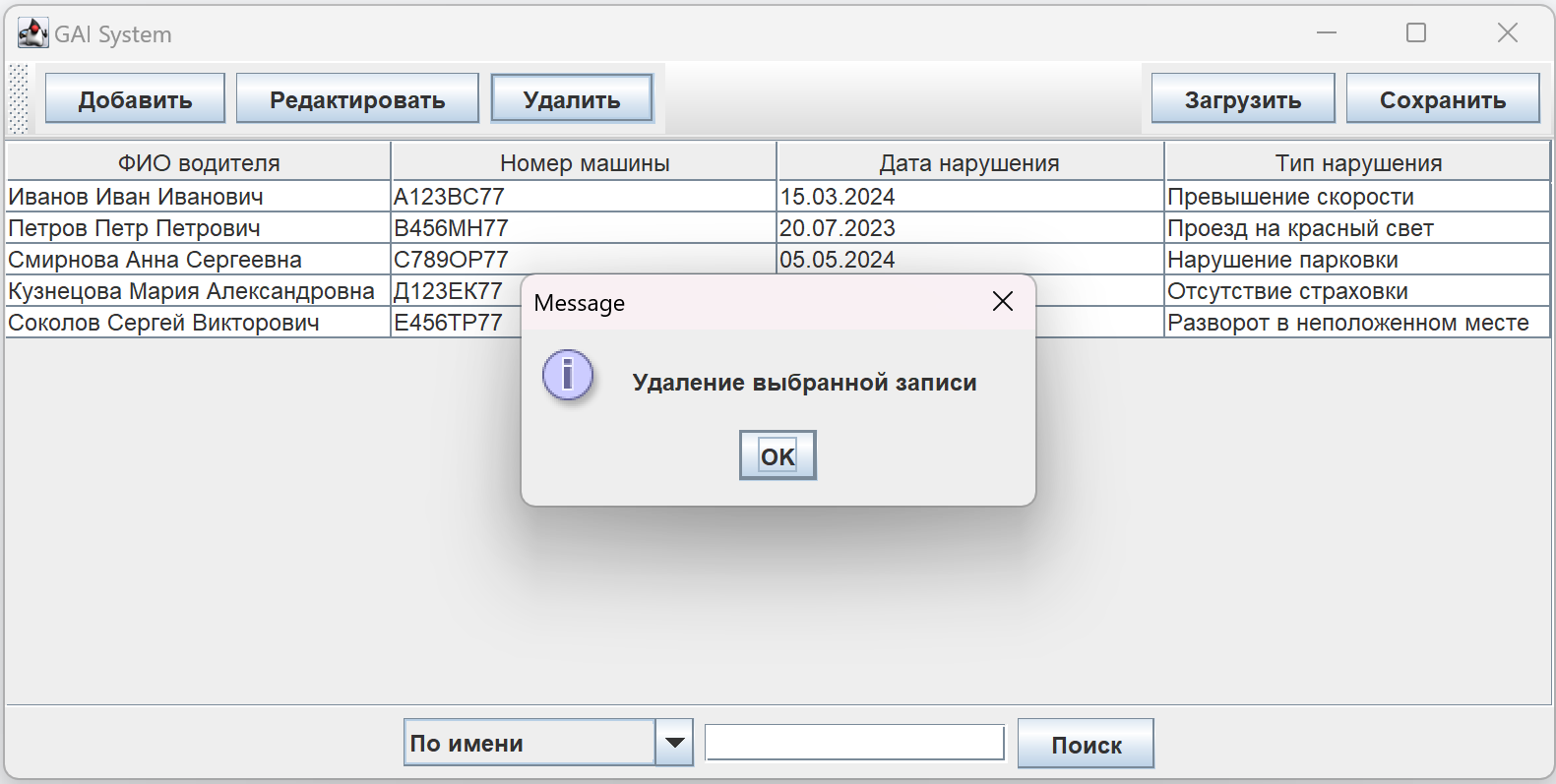


Рисунок 4 – Работа кнопки «Удалить»

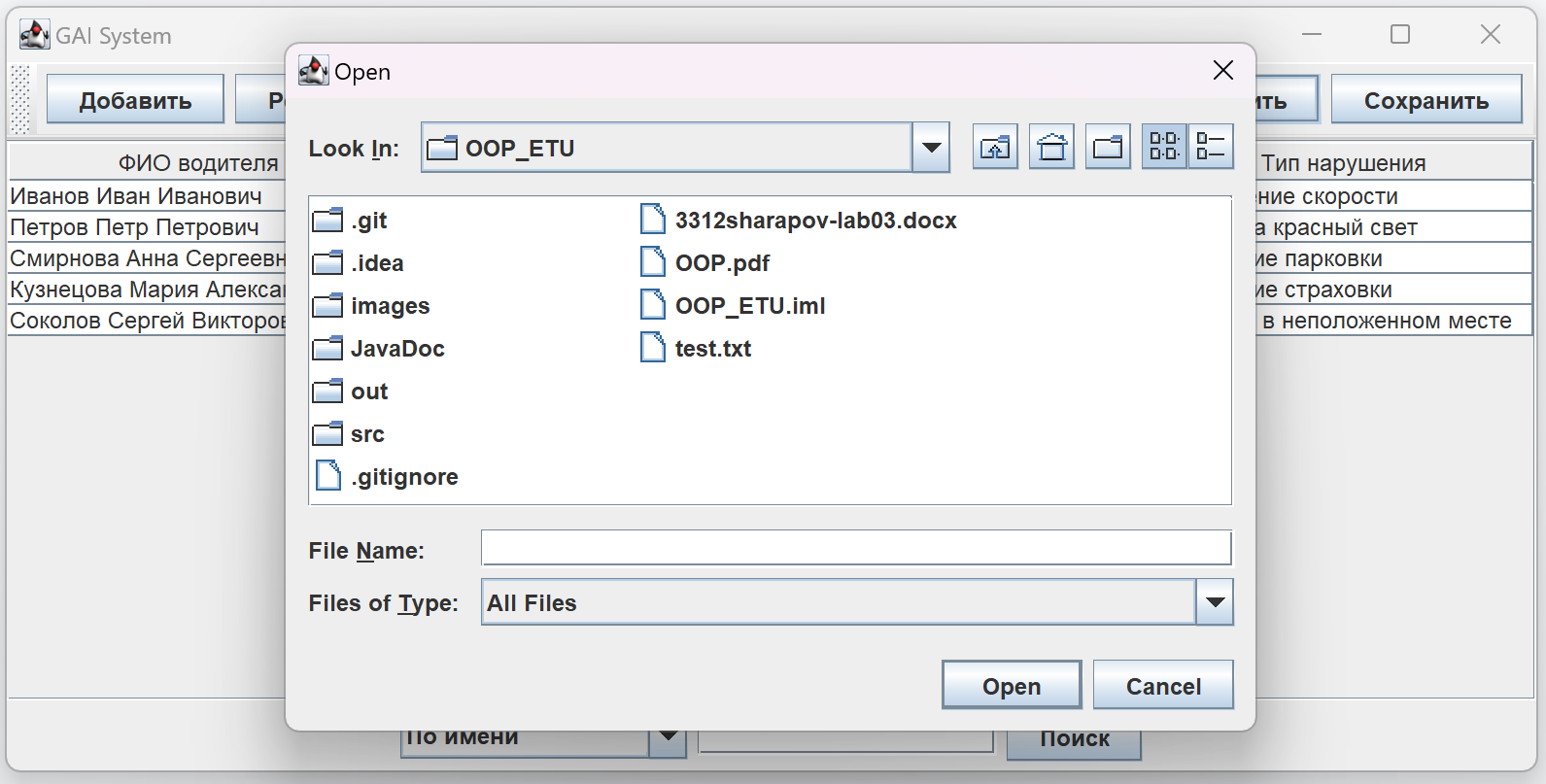


Рисунок 5 – Работа кнопок «Загрузить» и «Сохранить»

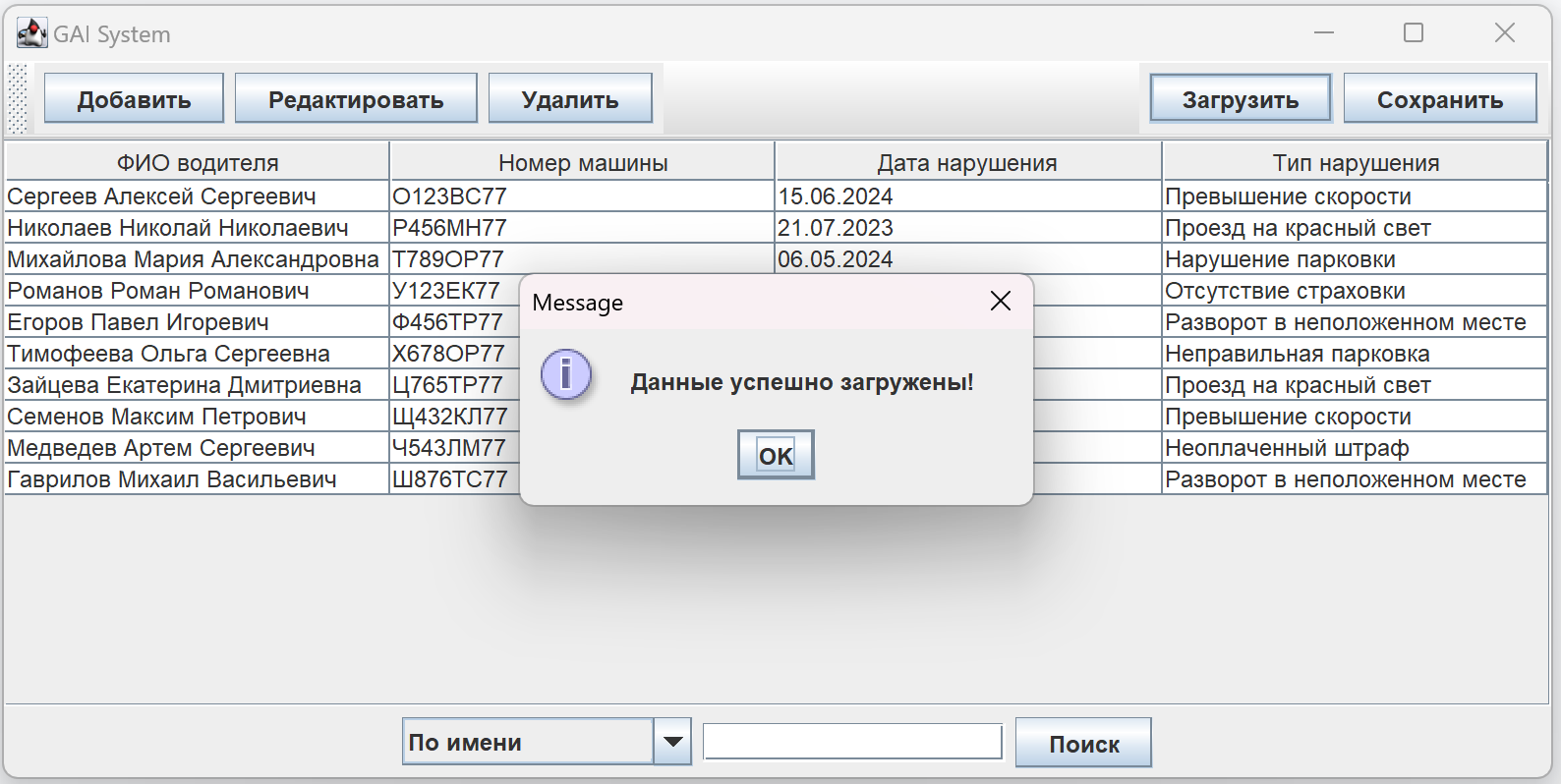


Рисунок 6 – Диалоговое окно после успешной загрузки данных

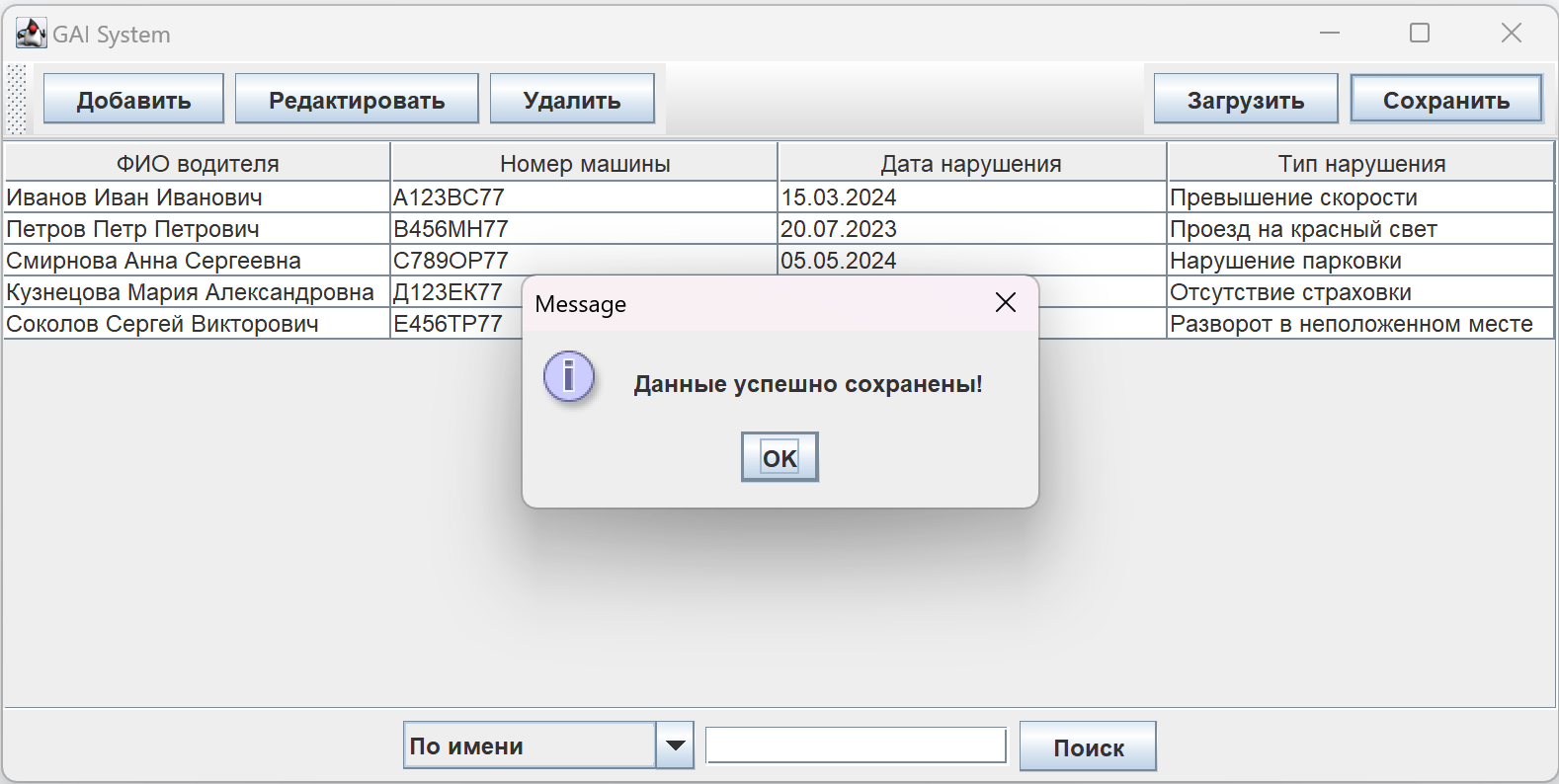


Рисунок 7 – Диалоговое окно после успешного сохранения данных

# Текст программы

|  |
| --- |
| **import javax.swing.\*; import javax.swing.table.DefaultTableModel; import java.awt.\*; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import java.io.\*;  /\*\*  \* Программа для работы с данными о водителях и их нарушениях.  \* Содержит функции добавления, редактирования, удаления записей, а также сохранения и загрузки данных в файл.  \*  \* @author Шарапов Иван 3312  \* @version 1.1  \*/ public class Main {  private JFrame mainFrame;  private DefaultTableModel tableModel;  private JTable dataTable;  private JButton addDriverButton, editDriverButton, deleteDriverButton, loadDriverButton, saveDriverButton;  private JTextField searchField;  private JComboBox<String> searchTypeComboBox;   /\*\*  \* Конструктор класса Main.  \* Инициализирует основное окно приложения для работы с данными.  \*/  public Main() {  // Конструктор по умолчанию, который инициализирует класс Main.  }   /\*\*  \* Метод для создания и отображения главного окна программы.  \* Включает создание таблицы, панели инструментов с кнопками и панель поиска.  \*/  public void show() {  // Создание основного окна приложения  mainFrame = new JFrame("GAI System");  mainFrame.setSize(800, 400);  mainFrame.setLocation(100, 100);  mainFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);   // Создание кнопок для управления записями  addDriverButton = new JButton("Добавить");  editDriverButton = new JButton("Редактировать");  deleteDriverButton = new JButton("Удалить");  loadDriverButton = new JButton("Загрузить");  saveDriverButton = new JButton("Сохранить");   // Панель инструментов, которая содержит кнопки  JToolBar toolBar = new JToolBar("Toolbar");  toolBar.setLayout(new BorderLayout()); // Устанавливаем BorderLayout для панели инструментов   // Панель с кнопками управления (добавление, редактирование, удаление)  JPanel leftPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));  leftPanel.add(addDriverButton);  leftPanel.add(editDriverButton);  leftPanel.add(deleteDriverButton);  toolBar.add(leftPanel, BorderLayout.WEST); // Размещаем в левой части панели   // Панель с кнопками сохранения и загрузки  JPanel rightPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));  rightPanel.add(loadDriverButton);  rightPanel.add(saveDriverButton);  toolBar.add(rightPanel, BorderLayout.EAST); // Размещаем в правой части панели   mainFrame.setLayout(new BorderLayout());  mainFrame.add(toolBar, BorderLayout.NORTH); // Размещение панели инструментов сверху   // Создание таблицы для отображения данных  String[] columns = {"ФИО водителя", "Номер машины", "Дата нарушения", "Тип нарушения"};  String[][] data = {  {"Иванов Иван Иванович", "А123ВС77", "15.03.2024", "Превышение скорости"},  {"Петров Петр Петрович", "В456МН77", "20.07.2023", "Проезд на красный свет"},  {"Смирнова Анна Сергеевна", "С789ОР77", "05.05.2024", "Нарушение парковки"},  {"Кузнецова Мария Александровна", "Д123ЕК77", "12.12.2023", "Отсутствие страховки"},  {"Соколов Сергей Викторович", "Е456ТР77", "22.02.2024", "Разворот в неположенном месте"}  };  tableModel = new DefaultTableModel(data, columns);  dataTable = new JTable(tableModel); // Таблица, которая использует данные tableModel  JScrollPane scrollPane = new JScrollPane(dataTable); // Добавляем прокрутку для таблицы  mainFrame.add(scrollPane, BorderLayout.CENTER); // Размещаем таблицу в центре окна   // Элементы поиска: поле ввода и кнопка "Поиск"  searchTypeComboBox = new JComboBox<>(new String[]{"По имени", "По номеру машины"});  searchField = new JTextField(15);  JButton searchButton = new JButton("Поиск");   JPanel searchPanel = new JPanel();  searchPanel.add(searchTypeComboBox);  searchPanel.add(searchField);  searchPanel.add(searchButton);  mainFrame.add(searchPanel, BorderLayout.SOUTH); // Панель поиска размещается внизу   // Добавляем действия для кнопок   // "Добавить" — действие для добавления новой записи  addDriverButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  JOptionPane.showMessageDialog(mainFrame, "Добавление новой записи");  }  });   // "Редактировать" — действие для редактирования выбранной записи  editDriverButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  JOptionPane.showMessageDialog(mainFrame, "Редактирование выбранной записи");  }  });   // "Удалить" — действие для удаления выбранной записи  deleteDriverButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  JOptionPane.showMessageDialog(mainFrame, "Удаление выбранной записи");  }  });   // "Загрузить" — открывает диалоговое окно для выбора файла и загружает данные в таблицу  loadDriverButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  JFileChooser fileChooser = new JFileChooser();  fileChooser.setCurrentDirectory(new File(System.getProperty("user.dir"))); // Устанавливаем корневую директорию проекта  int returnValue = fileChooser.showOpenDialog(mainFrame); // Показываем диалог открытия файла  if (returnValue == JFileChooser.APPROVE\_OPTION) { // Если файл выбран  File selectedFile = fileChooser.getSelectedFile(); // Получаем выбранный файл  loadDataFromFile(selectedFile); // Загружаем данные из файла в таблицу  }  }  });   // "Сохранить" — открывает диалоговое окно для сохранения файла и записывает данные таблицы в файл  saveDriverButton.addActionListener(new ActionListener() {  @Override  public void actionPerformed(ActionEvent e) {  JFileChooser fileChooser = new JFileChooser();  fileChooser.setCurrentDirectory(new File(System.getProperty("user.dir"))); // Устанавливаем корневую директорию проекта  int returnValue = fileChooser.showSaveDialog(mainFrame); // Показываем диалог сохранения файла  if (returnValue == JFileChooser.APPROVE\_OPTION) { // Если выбрано место для сохранения  File selectedFile = fileChooser.getSelectedFile(); // Получаем файл, который выбрал пользователь  saveDataToFile(selectedFile); // Сохраняем данные таблицы в этот файл  }  }  });   mainFrame.setVisible(true); // Делаем главное окно видимым  }   /\*\*  \* Сохраняет данные из таблицы в указанный файл.  \*  \* @param file Файл, в который будут сохранены данные.  \*/  private void saveDataToFile(File file) {  try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {  // Проходим по каждой строке таблицы  for (int row = 0; row < tableModel.getRowCount(); row++) {  for (int col = 0; col < tableModel.getColumnCount(); col++) {  // Записываем значения ячеек таблицы в файл через табуляцию  writer.write(tableModel.getValueAt(row, col).toString() + "\t");  }  writer.newLine(); // Переход на новую строку после каждой записи  }  JOptionPane.showMessageDialog(mainFrame, "Данные успешно сохранены!"); // Сообщение об успешном сохранении  } catch (IOException e) {  JOptionPane.showMessageDialog(mainFrame, "Ошибка при сохранении данных."); // Сообщение об ошибке  }  }   /\*\*  \* Загружает данные из указанного файла в таблицу.  \*  \* @param file Файл, из которого будут загружены данные.  \*/  private void loadDataFromFile(File file) {  try (BufferedReader reader = new BufferedReader(new FileReader(file))) {  String line;  tableModel.setRowCount(0); // Очищаем текущие данные таблицы перед загрузкой новых  while ((line = reader.readLine()) != null) {  // Разделяем строку по табуляциям, чтобы получить значения для каждой колонки  String[] rowData = line.split("\t");  tableModel.addRow(rowData); // Добавляем новую строку в таблицу  }  JOptionPane.showMessageDialog(mainFrame, "Данные успешно загружены!"); // Сообщение об успешной загрузке  } catch (IOException e) {  JOptionPane.showMessageDialog(mainFrame, "Ошибка при загрузке данных."); // Сообщение об ошибке  }  }   /\*\*  \* Точка входа в приложение. Запускает метод show() для отображения GUI.  \*  \* @param args Аргументы командной строки (не используются).  \*/  public static void main(String[] args) {  new Main().show(); // Запуск приложения  } }** |

# Приложение

Ссылка на видео: <https://rutube.ru/video/private/f32044acf724448233f195c4f3b617aa/?p=Iz6PfxGTQmcXmvYQYYQA7Q>

Ссылка на репозиторий: <https://github.com/DexTver/OOP_ETU/tree/lab_03>