

**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: Лебедев И.А.

Факультет: КТИ

Группа: №3312

Подпись преподавателя: _____

Санкт-Петербург

2024

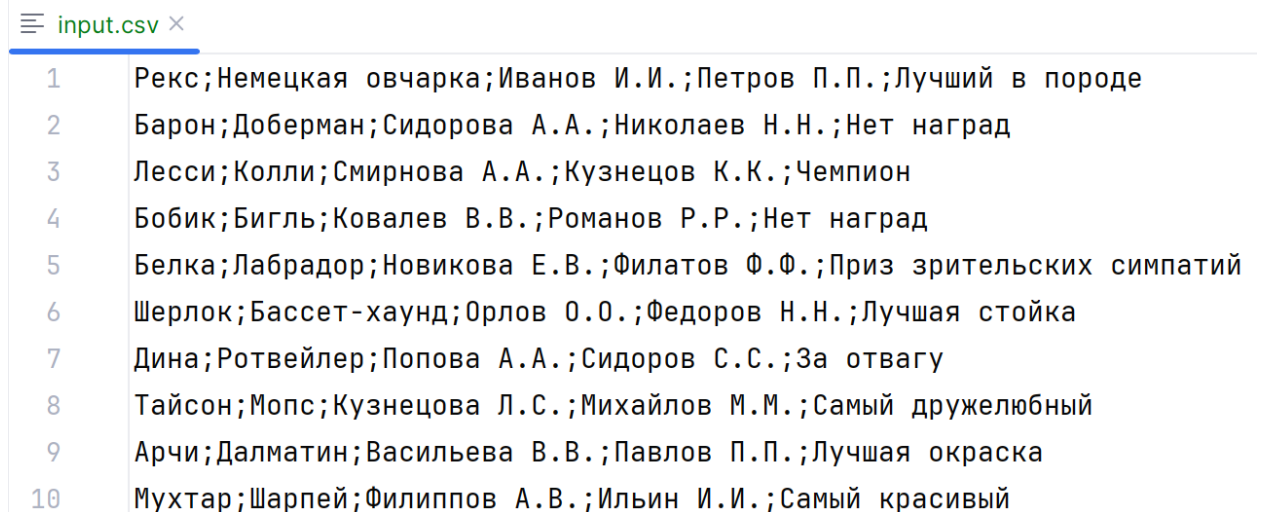
Содержание

Цель работы	3
Распечатки содержимого файлов с данными до и после внесения изменений	3
Скриншоты, иллюстрирующие процесс загрузки данных в файл и выгрузки из него	4
Текст программы	9
Приложение	16

Цель работы

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

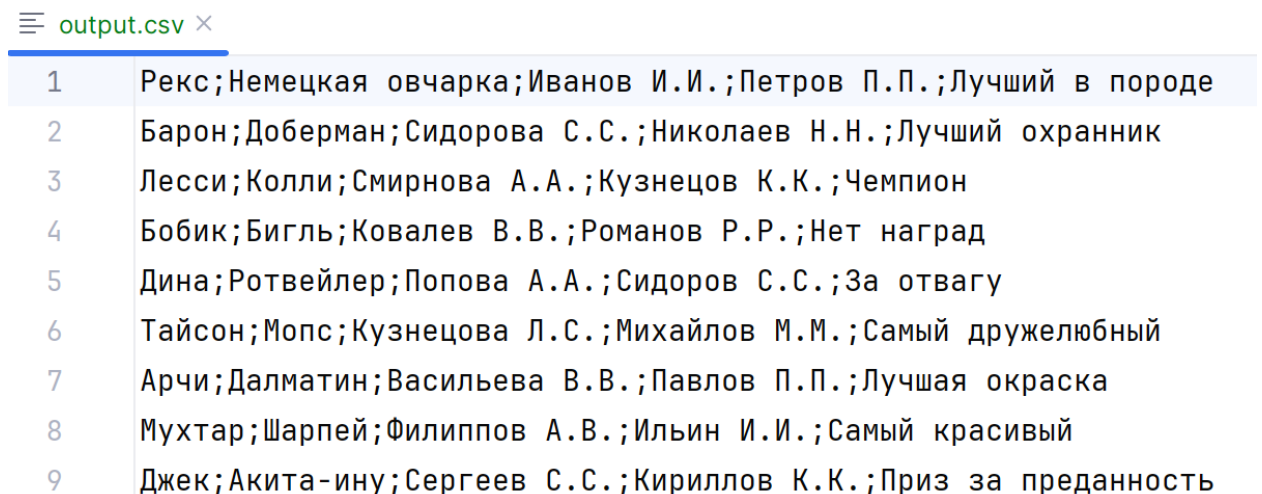
Распечатки содержимого файлов с данными до и после внесения изменений



The screenshot shows a text editor window titled 'input.csv'. It contains a list of 10 rows, each with a number in the first column and a semicolon-separated string of names and titles in the second column. The rows are: 1. Рекс;Немецкая овчарка;Иванов И.И.;Петров П.П.;Лучший в породе, 2. Барон;Доберман;Сидорова А.А.;Николаев Н.Н.;Нет наград, 3. Лесси;Колли;Смирнова А.А.;Кузнецов К.К.;Чемпион, 4. Бобик;Бигль;Ковалев В.В.;Романов Р.Р.;Нет наград, 5. Белка;Лабрадор;Новикова Е.В.;Филатов Ф.Ф.;Приз зрительских симпатий, 6. Шерлок;Бассет-хаунд;Орлов О.О.;Федоров Н.Н.;Лучшая стойка, 7. Дина;Ротвейлер;Попова А.А.;Сидоров С.С.;За отвагу, 8. Тайсон;Мопс;Кузнецова Л.С.;Михайлов М.М.;Самый дружелюбный, 9. Арчи;Далматин;Васильева В.В.;Павлов П.П.;Лучшая окраска, 10. Мухтар;Шарпей;Филиппов А.В.;Ильин И.И.;Самый красивый.

1	Рекс;Немецкая овчарка;Иванов И.И.;Петров П.П.;Лучший в породе
2	Барон;Доберман;Сидорова А.А.;Николаев Н.Н.;Нет наград
3	Лесси;Колли;Смирнова А.А.;Кузнецов К.К.;Чемпион
4	Бобик;Бигль;Ковалев В.В.;Романов Р.Р.;Нет наград
5	Белка;Лабрадор;Новикова Е.В.;Филатов Ф.Ф.;Приз зрительских симпатий
6	Шерлок;Бассет-хаунд;Орлов О.О.;Федоров Н.Н.;Лучшая стойка
7	Дина;Ротвейлер;Попова А.А.;Сидоров С.С.;За отвагу
8	Тайсон;Мопс;Кузнецова Л.С.;Михайлов М.М.;Самый дружелюбный
9	Арчи;Далматин;Васильева В.В.;Павлов П.П.;Лучшая окраска
10	Мухтар;Шарпей;Филиппов А.В.;Ильин И.И.;Самый красивый

Рисунок 1 – Содержимое исходного файла



The screenshot shows a text editor window titled 'output.csv'. It contains a list of 9 rows, each with a number in the first column and a semicolon-separated string of names and titles in the second column. The rows are: 1. Рекс;Немецкая овчарка;Иванов И.И.;Петров П.П.;Лучший в породе, 2. Барон;Доберман;Сидорова С.С.;Николаев Н.Н.;Лучший охранник, 3. Лесси;Колли;Смирнова А.А.;Кузнецов К.К.;Чемпион, 4. Бобик;Бигль;Ковалев В.В.;Романов Р.Р.;Нет наград, 5. Дина;Ротвейлер;Попова А.А.;Сидоров С.С.;За отвагу, 6. Тайсон;Мопс;Кузнецова Л.С.;Михайлов М.М.;Самый дружелюбный, 7. Арчи;Далматин;Васильева В.В.;Павлов П.П.;Лучшая окраска, 8. Мухтар;Шарпей;Филиппов А.В.;Ильин И.И.;Самый красивый, 9. Джек;Акита-ину;Сергеев С.С.;Кириллов К.К.;Приз за преданность.

1	Рекс;Немецкая овчарка;Иванов И.И.;Петров П.П.;Лучший в породе
2	Барон;Доберман;Сидорова С.С.;Николаев Н.Н.;Лучший охранник
3	Лесси;Колли;Смирнова А.А.;Кузнецов К.К.;Чемпион
4	Бобик;Бигль;Ковалев В.В.;Романов Р.Р.;Нет наград
5	Дина;Ротвейлер;Попова А.А.;Сидоров С.С.;За отвагу
6	Тайсон;Мопс;Кузнецова Л.С.;Михайлов М.М.;Самый дружелюбный
7	Арчи;Далматин;Васильева В.В.;Павлов П.П.;Лучшая окраска
8	Мухтар;Шарпей;Филиппов А.В.;Ильин И.И.;Самый красивый
9	Джек;Акита-ину;Сергеев С.С.;Кириллов К.К.;Приз за преданность

Рисунок 2 – Содержимое файла с данными после изменений

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

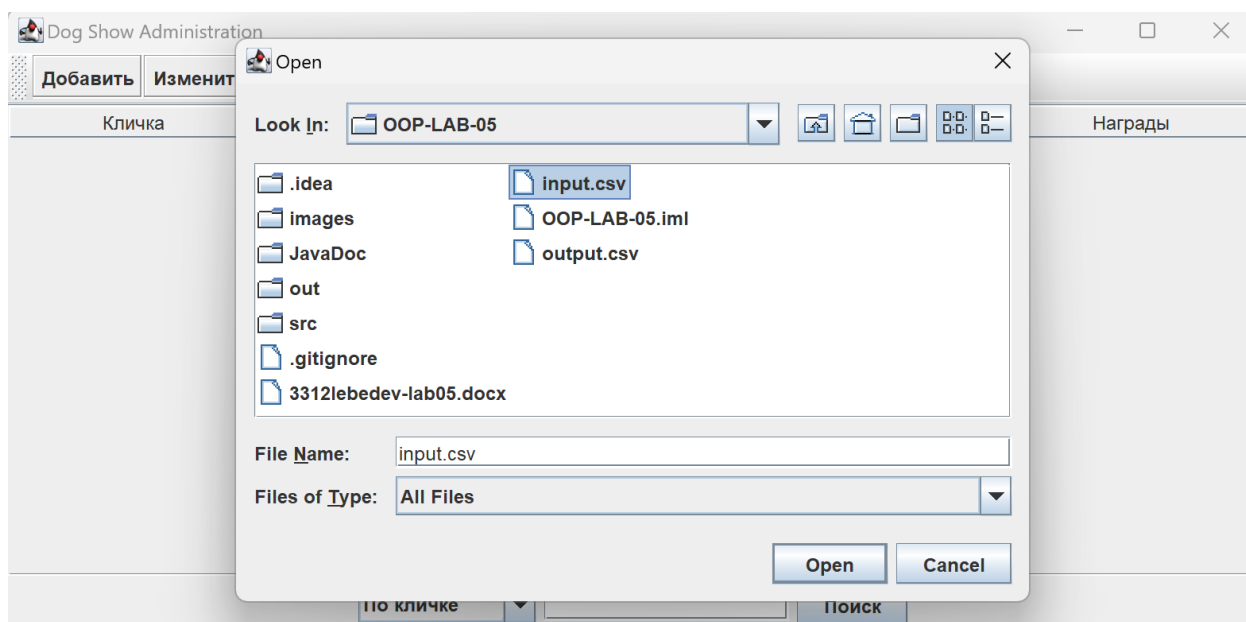


Рисунок 3 – Загрузка данных из исходного файла

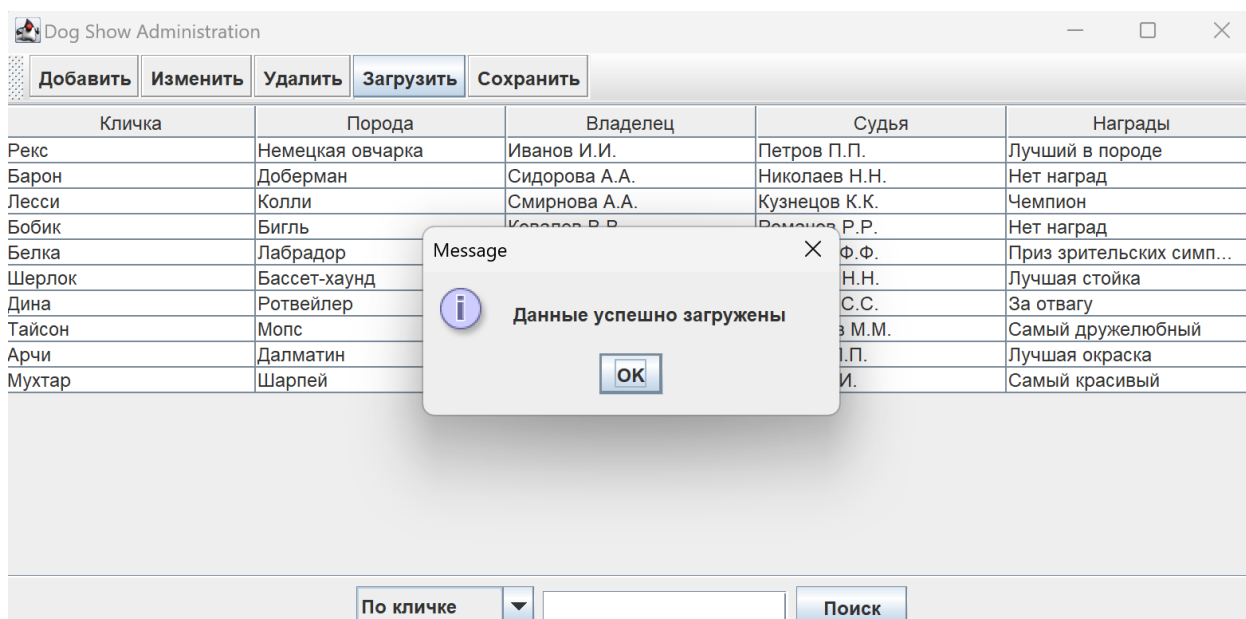


Рисунок 4 – Успешная загрузка данных из файла *input.csv*

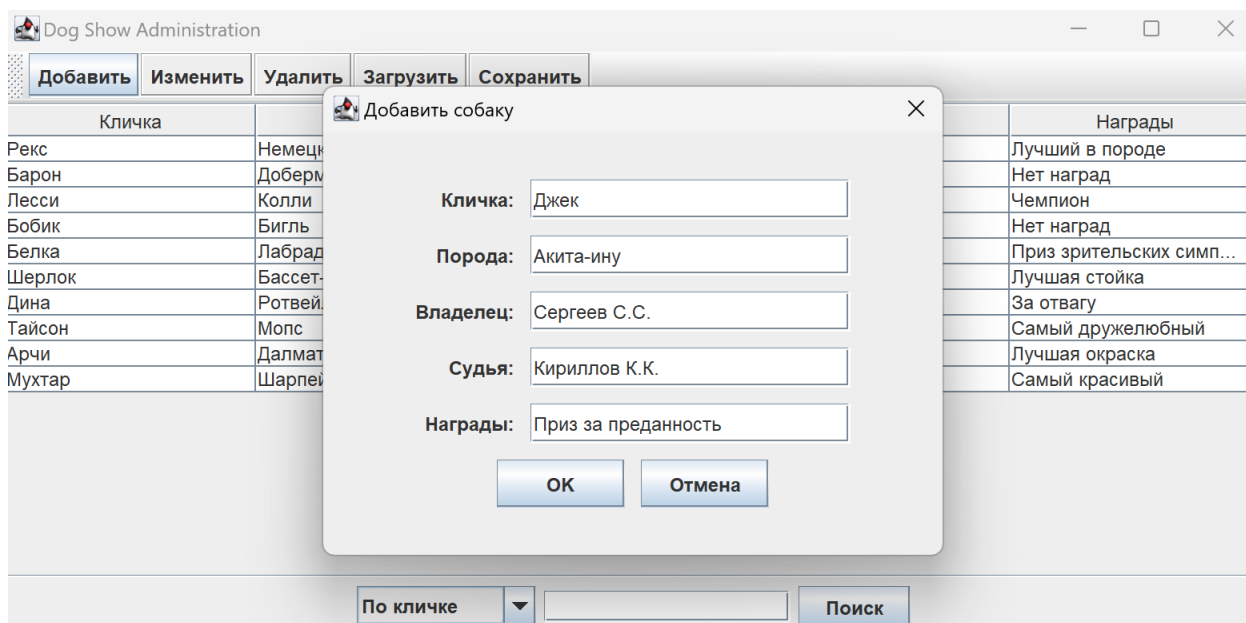


Рисунок 5 – Добавление данных о новой собаке

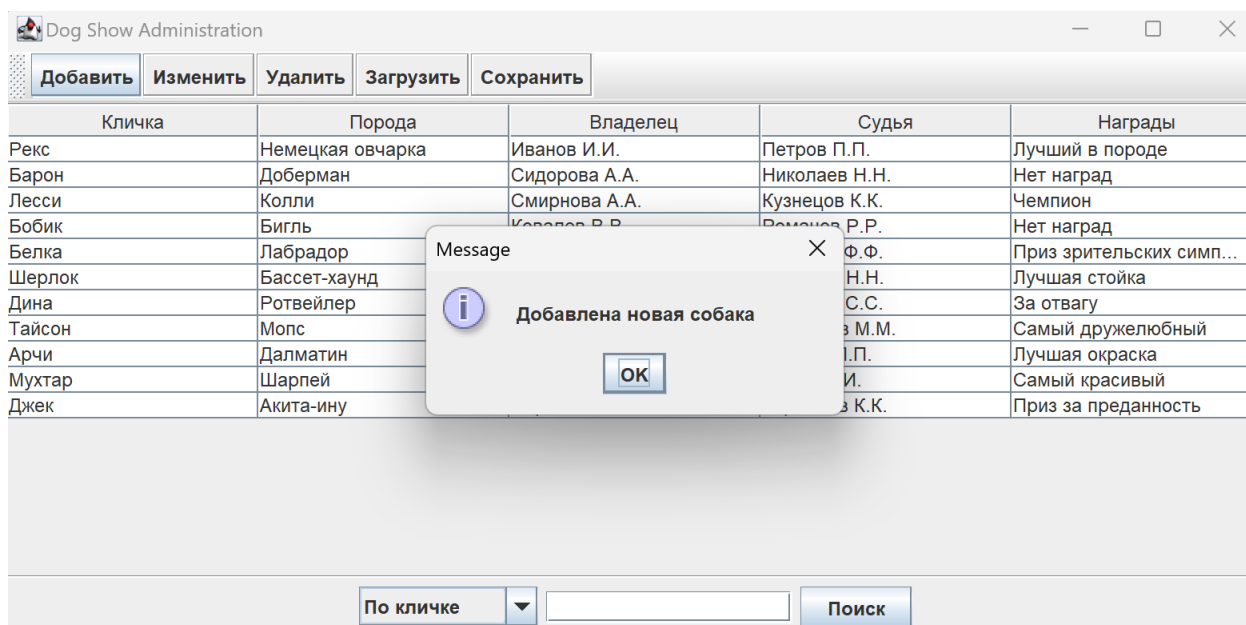


Рисунок 6 – Успешное добавление новой собаки

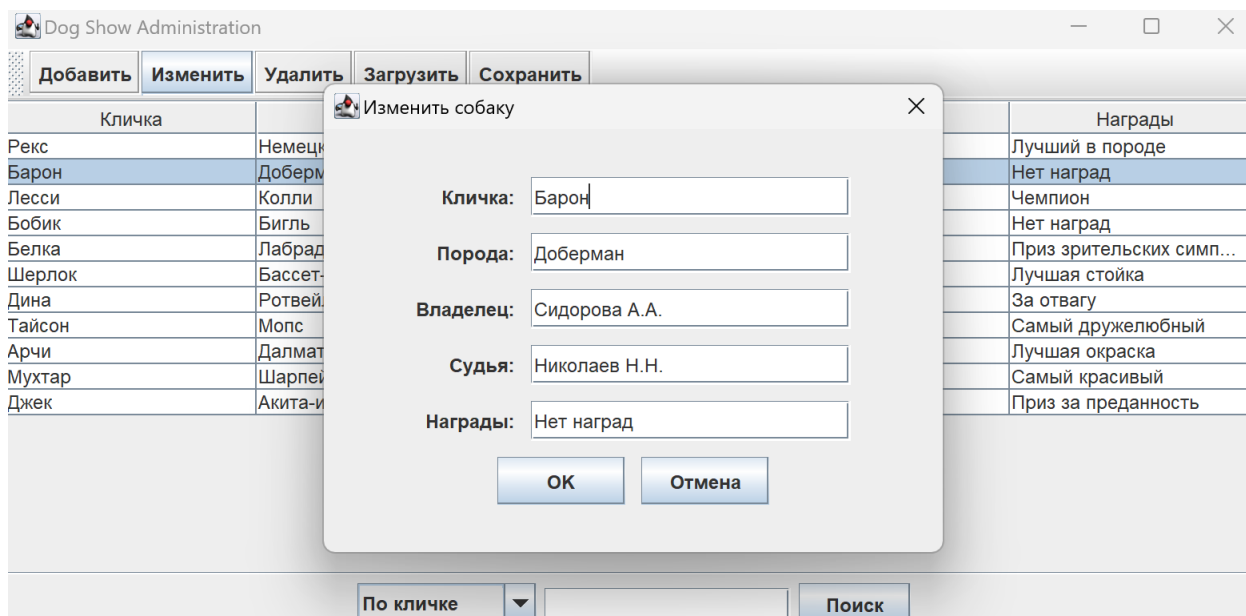


Рисунок 7 – Данные существующей собаки до их изменения

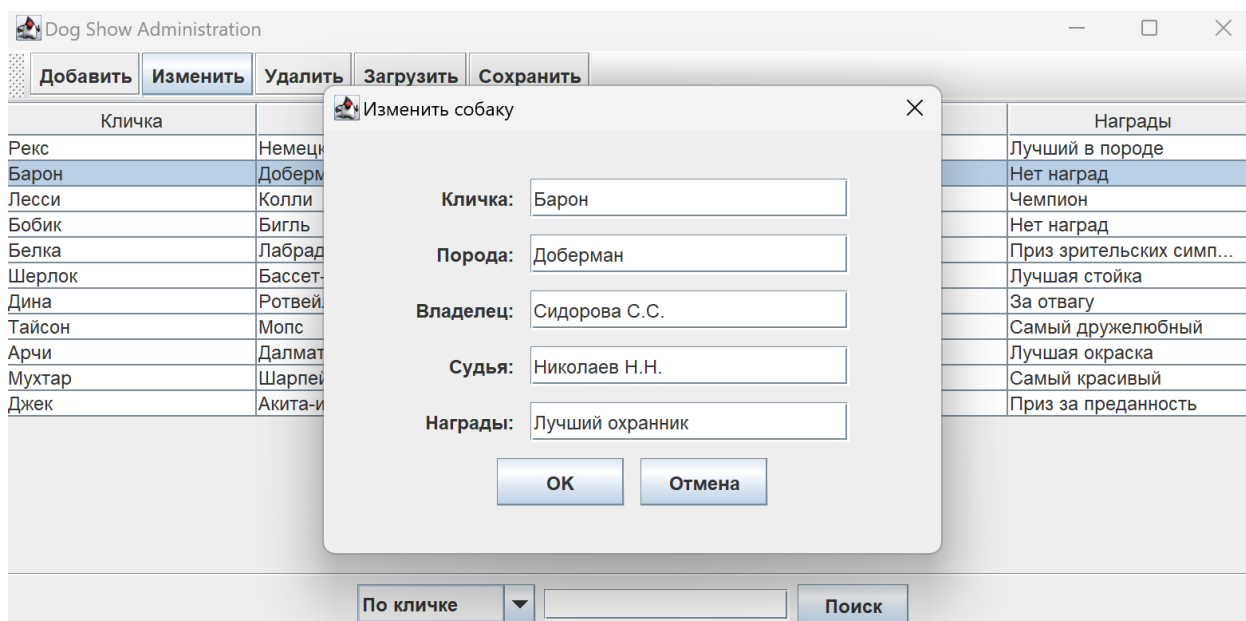


Рисунок 8 – Изменение данных уже существующей собаки

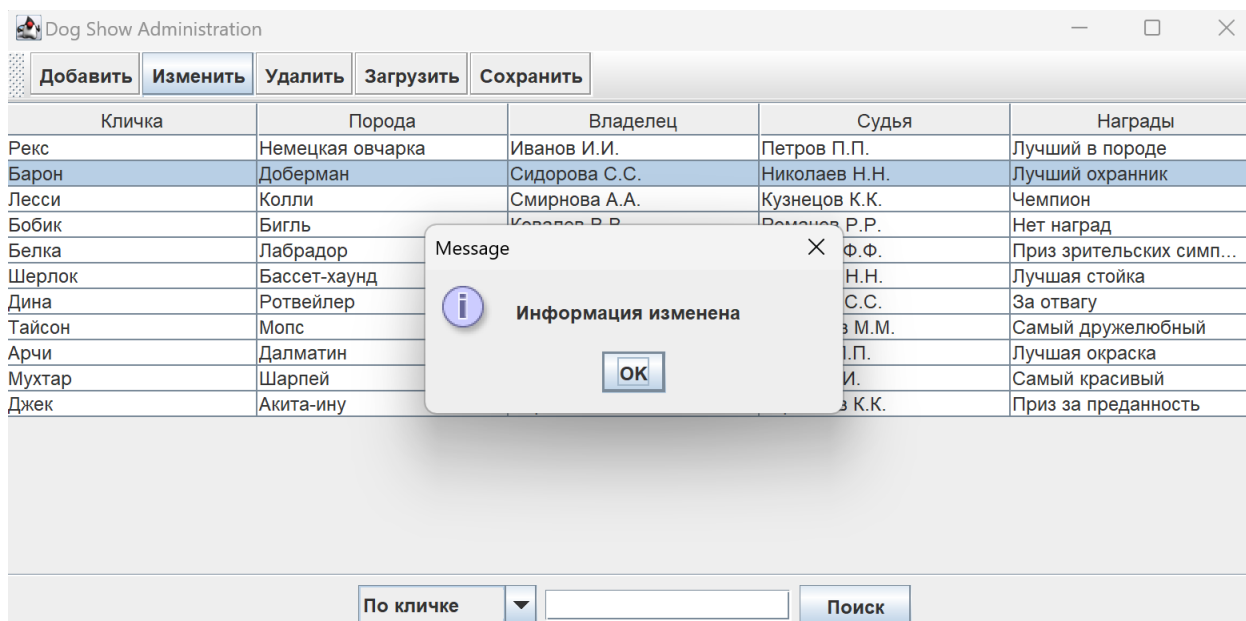


Рисунок 9 – Успешное изменение информации о собаке

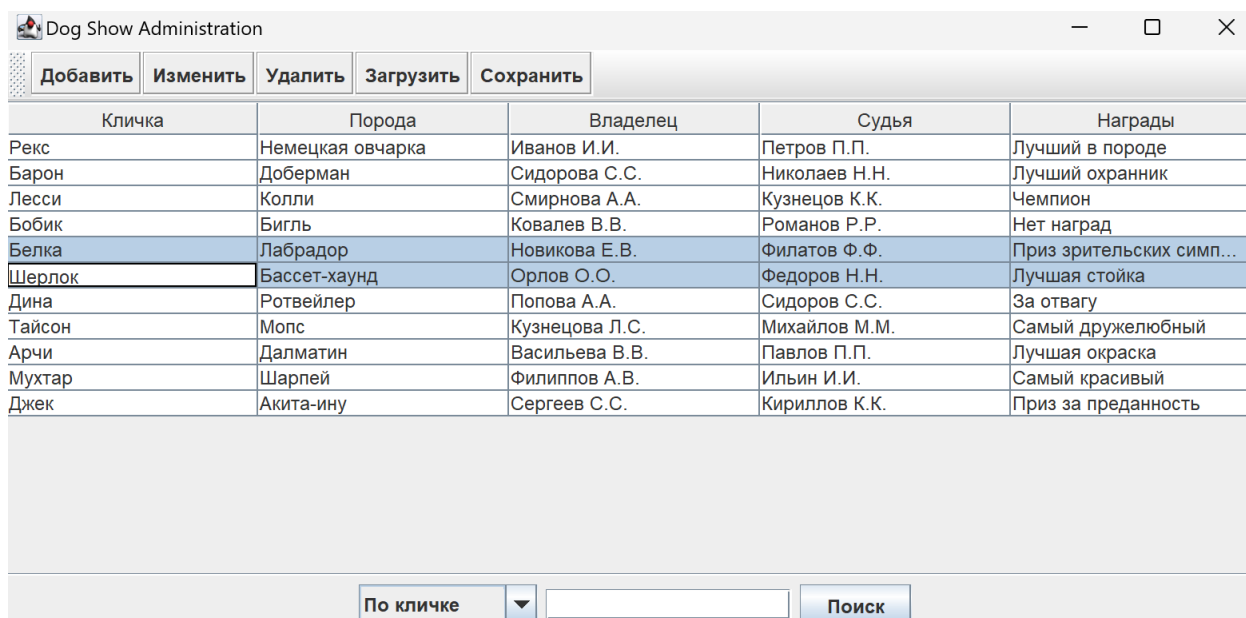


Рисунок 10 – Выделение собак для последующего удаления

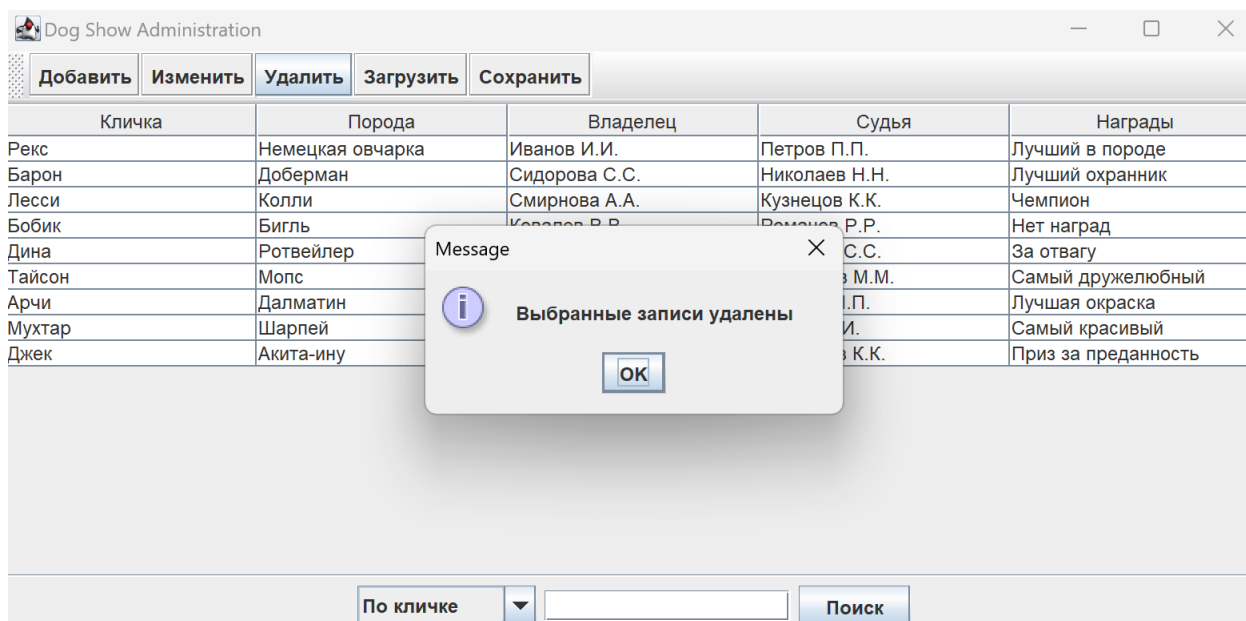


Рисунок 11 – Успешное удаление выделенных собак

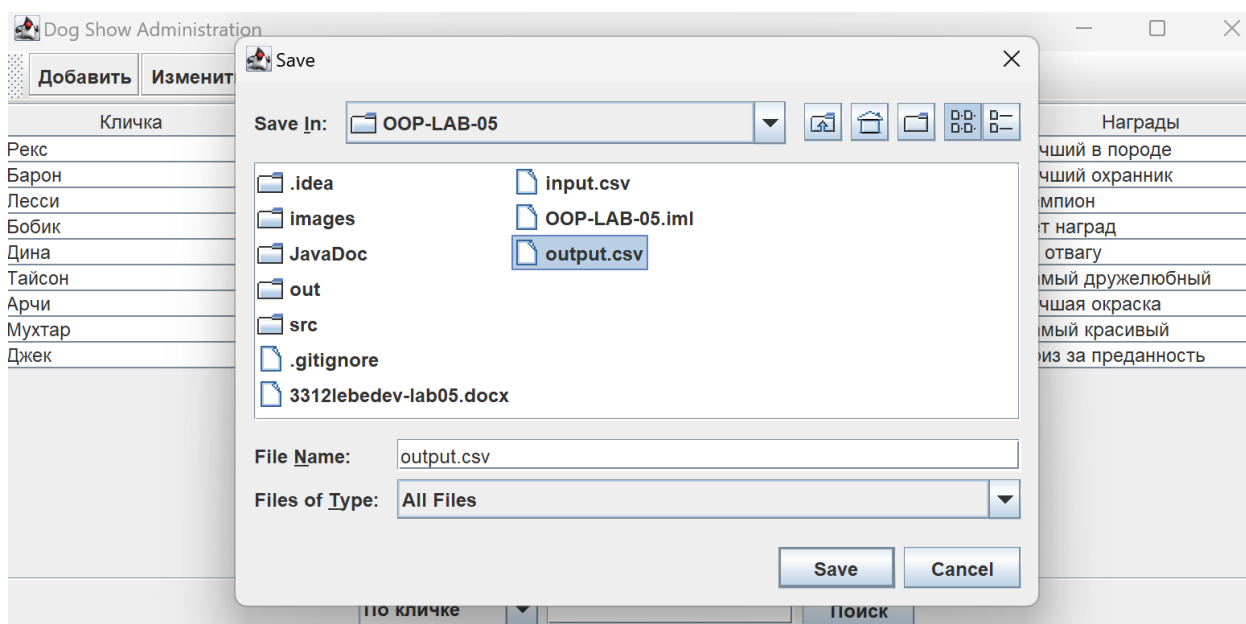


Рисунок 12 – Сохранение данных в результирующий файл после изменений

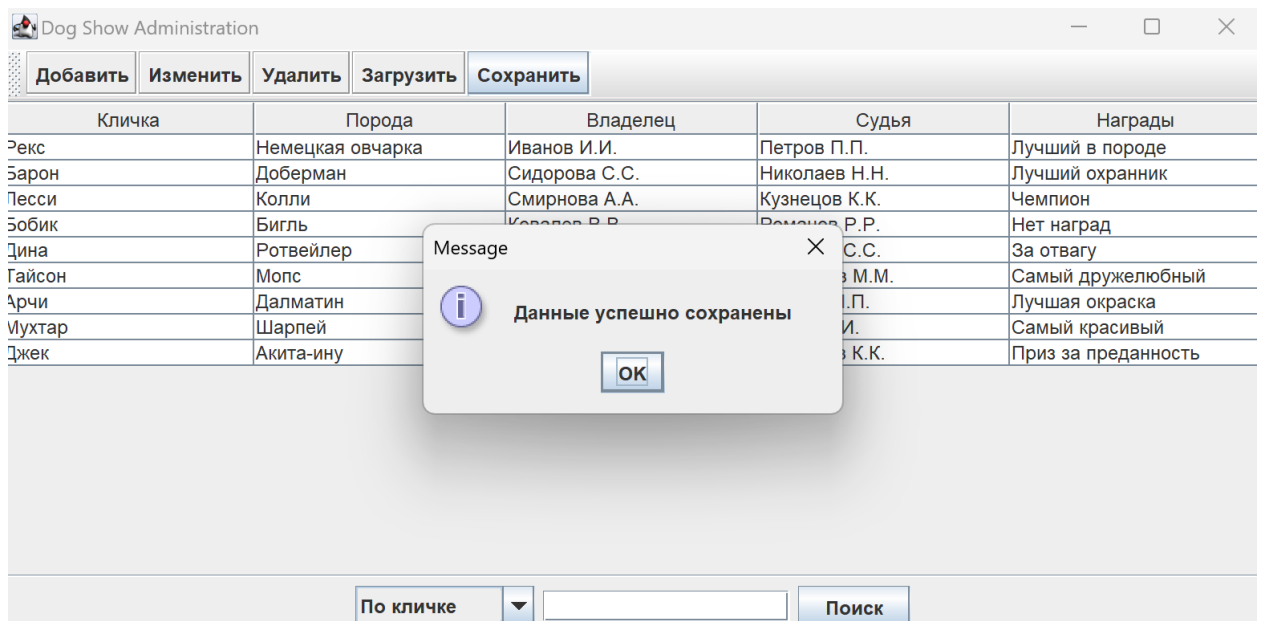


Рисунок 13 – Успешное сохранение данных в файл *output.csv*

Текст программы

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.*;
import java.io.*;

/**
 * Исключение, выбрасываемое при попытке выполнить действие без выбора строки.
 */
class InvalidSelectionException extends Exception {
    public InvalidSelectionException(String message) {
        super(message);
    }
}

/**
 * @author Лебедев Игнат 3312
 * @version 1.0
 */
public class Main {
    private JFrame mainFrame;
    private DefaultTableModel tableModel;
    private JTable dataTable;
    private JButton addDogButton, editDogButton, deleteDogButton, loadDogButton,
    saveDogButton;
    private JTextField searchField;
    private JComboBox<String> searchCriteriaComboBox;
    private boolean unsavedChanges = false;

    /**
     * Метод для построения и визуализации экранной формы.
     */
    public void show() {
        mainFrame = new JFrame("Dog Show Administration");
        mainFrame.setSize(800, 400);
    }
}
```

```

mainFrame.setLocation(100, 100);
mainFrame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

JPanel mainPanel = new JPanel(new BorderLayout());

// Создание кнопок
addDogButton = new JButton("Добавить");
editDogButton = new JButton("Изменить");
deleteDogButton = new JButton("Удалить");
loadDogButton = new JButton("Загрузить");
saveDogButton = new JButton("Сохранить");

// Панель инструментов с кнопками
JToolBar toolBar = new JToolBar("Панель инструментов");
toolBar.add(addDogButton);
toolBar.add(editDogButton);
toolBar.add(deleteDogButton);
toolBar.add(loadDogButton);
toolBar.add(saveDogButton);

mainPanel.add(toolBar, BorderLayout.NORTH);

// Данные для таблицы
String[] columns = {"Кличка", "Порода", "Владелец", "Судья", "Награды"};
TableModel tableModel = new DefaultTableModel(columns, 0);
JTable dataTable = new JTable(tableModel);
dataTable.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
// Позволяем множественный выбор строк
JScrollPane scrollPane = new JScrollPane(dataTable);

mainPanel.add(scrollPane, BorderLayout.CENTER);

// Панель поиска
JPanel searchPanel = new JPanel();
searchCriteriaComboBox = new JComboBox<>(new String[]{"По кличке", "По породе", "По владельцу", "По судье", "По награде"});
searchField = new JTextField(15);
JButton searchButton = new JButton("Поиск");

searchPanel.add(searchCriteriaComboBox);
searchPanel.add(searchField);
searchPanel.add(searchButton);

mainPanel.add(searchPanel, BorderLayout.SOUTH);
mainFrame.add(mainPanel);

// Логика для кнопки "Поиск"
searchButton.addActionListener(e -> performSearch());

// Логика для кнопки "Добавить"
addDogButton.addActionListener(e -> {
    DogEntryDialog dialog = new DogEntryDialog(mainFrame, "Добавить собаку", null);
    dialog.setVisible(true);
    String[] newData = dialog.getDogData();
    if (newData != null) {
        tableModel.addRow(newData);
        unsavedChanges = true;
        JOptionPane.showMessageDialog(mainFrame, "Добавлена новая собака");
    }
}

```

```

    });

    // Логика для кнопки "Изменить"
    editDogButton.addActionListener(e -> {
        try {
            validateSelectionForEdit(dataTable);
            int selectedRow = dataTable.getSelectedRow();
            String[] currentData = new String[tableModel.getColumnCount()];
            for (int i = 0; i < tableModel.getColumnCount(); i++) {
                currentData[i] = (String) tableModel.getValueAt(selectedRow,
i);
            }
            DogEntryDialog dialog = new DogEntryDialog(mainFrame, "Изменить
собаку", currentData);
            dialog.setVisible(true);
            String[] updatedData = dialog.getDogData();
            if (updatedData != null) {
                for (int i = 0; i < updatedData.length; i++) {
                    tableModel.setValueAt(updatedData[i], selectedRow, i);
                }
                unsavedChanges = true;
                JOptionPane.showMessageDialog(mainFrame, "Информация
изменена");
            }
        } catch (InvalidSelectionException ex) {
            JOptionPane.showMessageDialog(mainFrame, ex.getMessage());
        }
    });

    // Логика для кнопки "Удалить"
    deleteDogButton.addActionListener(e -> {
        try {
            validateSelection(dataTable);
            int[] selectedRows = dataTable.getSelectedRows();

            if (selectedRows.length == 0) {
                throw new InvalidSelectionException("Не выбраны строки для
удаления.");
            }

            // Удаляем выбранные строки, начиная с конца, чтобы не нарушить
индексы
            for (int i = selectedRows.length - 1; i >= 0; i--) {
                tableModel.removeRow(selectedRows[i]);
            }
            unsavedChanges = true;
            JOptionPane.showMessageDialog(mainFrame, "Выбранные записи
удалены");
        } catch (InvalidSelectionException ex) {
            JOptionPane.showMessageDialog(mainFrame, ex.getMessage());
        }
    });

    // Реализация кнопки "Загрузить"
    loadDogButton.addActionListener(e -> loadDataFromFile());

    // Реализация кнопки "Сохранить"
    saveDogButton.addActionListener(e -> saveDataToFile());

```

```

// Обработка закрытия окна с проверкой на несохраненные изменения
mainFrame.addWindowListener(new WindowAdapter() {
    @Override
    public void windowClosing(WindowEvent e) {
        if (unsavedChanges) {
            int response = JOptionPane.showConfirmDialog(mainFrame, "Есть
несохраненные изменения. Хотите сохранить перед выходом?", "Несохраненные
изменения", JOptionPane.YES_NO_CANCEL_OPTION);
            if (response == JOptionPane.YES_OPTION) {
                saveDataToFile();
                mainFrame.dispose();
            } else if (response == JOptionPane.NO_OPTION) {
                mainFrame.dispose();
            }
        } else {
            mainFrame.dispose();
        }
    }
});

mainFrame.setVisible(true);
}

/**
 * Выполняет поиск по выбранному критерию и подсвечивает все строки с
 * совпадениями.
 */
private void performSearch() {
    String searchText = searchField.getText().trim().toLowerCase();
    if (searchText.isEmpty()) {
        JOptionPane.showMessageDialog(mainFrame, "Введите текст для поиска.");
        return;
    }

    int searchColumn = searchCriteriaComboBox.getSelectedIndex();
    dataTable.clearSelection(); // Снимаем выделение перед поиском

    boolean found = false;
    for (int i = 0; i < tableModel.getRowCount(); i++) {
        String cellValue = tableModel.getValueAt(i,
searchColumn).toString().toLowerCase();
        if (cellValue.contains(searchText)) {
            dataTable.addRowSelectionInterval(i, i); // Подсвечиваем строку
            found = true;
        }
    }

    if (!found) {
        JOptionPane.showMessageDialog(mainFrame, "Совпадения не найдены.");
    }
}

/**
 * Метод для загрузки данных из файла в таблицу.
 */
private void loadDataFromFile() {
    JFileChooser fileChooser = new JFileChooser();
    fileChooser.setCurrentDirectory(new File(System.getProperty("user.dir")));
    int result = fileChooser.showOpenDialog(mainFrame);

```

```

        if (result == JFileChooser.APPROVE_OPTION) {
            File file = fileChooser.getSelectedFile();
            try (BufferedReader reader = new BufferedReader(new FileReader(file)))
            {
                tableModel.setRowCount(0); // Очистка таблицы перед загрузкой
                данных
                String line;
                while ((line = reader.readLine()) != null) {
                    String[] rowData = line.split(";");
                    tableModel.addRow(rowData);
                }
                unsavedChanges = true;
                JOptionPane.showMessageDialog(mainFrame, "Данные успешно
загружены");
            }
            catch (IOException e) {
                JOptionPane.showMessageDialog(mainFrame, "Ошибка при загрузке
данных");
            }
        }
    }

    /**
     * Метод для сохранения данных из таблицы в файл.
     */
    private void saveDataToFile() {
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setCurrentDirectory(new File(System.getProperty("user.dir")));
        int result = fileChooser.showSaveDialog(mainFrame);

        if (result == JFileChooser.APPROVE_OPTION) {
            File file = fileChooser.getSelectedFile();
            try (BufferedWriter writer = new BufferedWriter(new FileWriter(file)))
            {
                for (int i = 0; i < tableModel.getRowCount(); i++) {
                    for (int j = 0; j < tableModel.getColumnCount(); j++) {
                        writer.write((String) tableModel.getValueAt(i, j));
                        if (j < tableModel.getColumnCount() - 1) {
                            writer.write(";");
                        }
                    }
                    writer.newLine();
                }
                unsavedChanges = false;
                JOptionPane.showMessageDialog(mainFrame, "Данные успешно
сохранены");
            }
            catch (IOException e) {
                JOptionPane.showMessageDialog(mainFrame, "Ошибка при сохранении
данных");
            }
        }
    }

    /**
     * Метод проверки, выбрана ли строка в таблице для удаления.
     * @throws InvalidSelectionException если строка не выбрана
     */
    private void validateSelection(JTable table) throws InvalidSelectionException {
        if (table.getSelectedRowCount() == 0) {

```

```

        throw new InvalidSelectionException("Не выбраны строки для удаления.");
    }
}

/**
 * Метод проверки, выбрана ли строка в таблице для изменения.
 * @throws InvalidSelectionException если строка не выбрана
 */
private void validateSelectionForEdit(JTable table) throws
InvalidSelectionException {
    if (table.getSelectedRow() == -1) {
        throw new InvalidSelectionException("Не выбрана строка для
изменения.");
    }
}

/**
 * Диалоговое окно для добавления или редактирования записи о собаке.
 */
private static class DogEntryDialog extends JDialog {
    private JTextField nameField, breedField, ownerField, judgeField,
awardField;
    private String[] dogData;

    public DogEntryDialog(JFrame parent, String title, String[] currentData) {
        super(parent, title, true);
        setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(5, 5, 5, 5);

        Dimension fieldSize = new Dimension(200, 25);

        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.anchor = GridBagConstraints.EAST;
        add(new JLabel("Кличка:"), gbc);
        nameField = new JTextField(currentData == null ? "" : currentData[0]);
        nameField.setPreferredSize(fieldSize);
        gbc.gridx = 1;
        gbc.anchor = GridBagConstraints.WEST;
        add(nameField, gbc);

        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.anchor = GridBagConstraints.EAST;
        add(new JLabel("Порода:"), gbc);
        breedField = new JTextField(currentData == null ? "" : currentData[1]);
        breedField.setPreferredSize(fieldSize);
        gbc.gridx = 1;
        gbc.anchor = GridBagConstraints.WEST;
        add(breedField, gbc);

        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.anchor = GridBagConstraints.EAST;
        add(new JLabel("Владелец:"), gbc);
        ownerField = new JTextField(currentData == null ? "" : currentData[2]);
        ownerField.setPreferredSize(fieldSize);
        gbc.gridx = 1;
        gbc.anchor = GridBagConstraints.WEST;

```

```

        add(ownerField, gbc);

        gbc.gridx = 0;
        gbc.gridy = 3;
        gbc.anchor = GridBagConstraints.EAST;
        add(new JLabel("Судья:"), gbc);
        judgeField = new JTextField(currentData == null ? "" : currentData[3]);
        judgeField.setPreferredSize(fieldSize);
        gbc.gridx = 1;
        gbc.anchor = GridBagConstraints.WEST;
        add(judgeField, gbc);

        gbc.gridx = 0;
        gbc.gridy = 4;
        gbc.anchor = GridBagConstraints.EAST;
        add(new JLabel("Награды:"), gbc);
        awardField = new JTextField(currentData == null ? "" : currentData[4]);
        awardField.setPreferredSize(fieldSize);
        gbc.gridx = 1;
        gbc.anchor = GridBagConstraints.WEST;
        add(awardField, gbc);

        // Центральное расположение кнопок "ОК" и "Отмена"
        JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10,
0));

        JButton confirmButton = new JButton("ОК");
        confirmButton.setPreferredSize(new Dimension(80, 30));
        JButton cancelButton = new JButton("Отмена");
        cancelButton.setPreferredSize(new Dimension(80, 30));
        buttonPanel.add(confirmButton);
        buttonPanel.add(cancelButton);

        gbc.gridx = 0;
        gbc.gridy = 5;
        gbc.gridwidth = 2;
        gbc.anchor = GridBagConstraints.CENTER;
        add(buttonPanel, gbc);

        confirmButton.addActionListener(e -> onSave());
        cancelButton.addActionListener(e -> onCancel());

        setSize(400, 300);
        setLocationRelativeTo(parent);
    }

    private void onSave() {
        if (nameField.getText().isEmpty() || breedField.getText().isEmpty() ||
            ownerField.getText().isEmpty() ||
judgeField.getText().isEmpty() ||
            awardField.getText().isEmpty()) {
            JOptionPane.showMessageDialog(this, "Все поля должны быть
заполнены.");
        } else {
            dogData = new String[]{
                nameField.getText(),
                breedField.getText(),
                ownerField.getText(),
                judgeField.getText(),
                awardField.getText()
            };
        }
    }

```

```

        setVisible(false);
    }
}

/**
 * Отменяет операцию и скрывает форму.
 */
private void onCancel() {
    dogData = null;
    setVisible(false);
}

/**
 * Возвращает данные собаки.
 * @return массив строк с данными собаки или null, если данных нет.
 */
public String[] getDogData() {
    return dogData;
}

}

/**
 * Главный метод для запуска приложения.
 * @param args аргументы командной строки
 */
public static void main(String[] args) {
    new Main().show();
}
}

```

Приложение

Ссылка на репозиторий: <https://github.com/TrueTalentless/OOP-LAB-05>