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 № 7
«Построение отчётов в PDF- и HTML- форматах»
по дисциплине «Объектно - ориентированное программирование на языке Java»

Выполнил: Лебедев И.А.

Факультет: КТИ

Группа: №3312

Подпись преподавателя: _____

Санкт-Петербург

2024

Содержание

Цель работы

Знакомство со способами формирования отчётов с использованием конструктора Jaspersoft Studio.

Построение шаблона

Создадим новый отчёт, для этого в Jaspersoft Studio нажмём File → New → Jasper Report. Далее выберем шаблон (Рис.1) и нажмём кнопку Next. Далее выберем имя и место сохранения нашего шаблона (Рис. 2) нажмём кнопку Next.

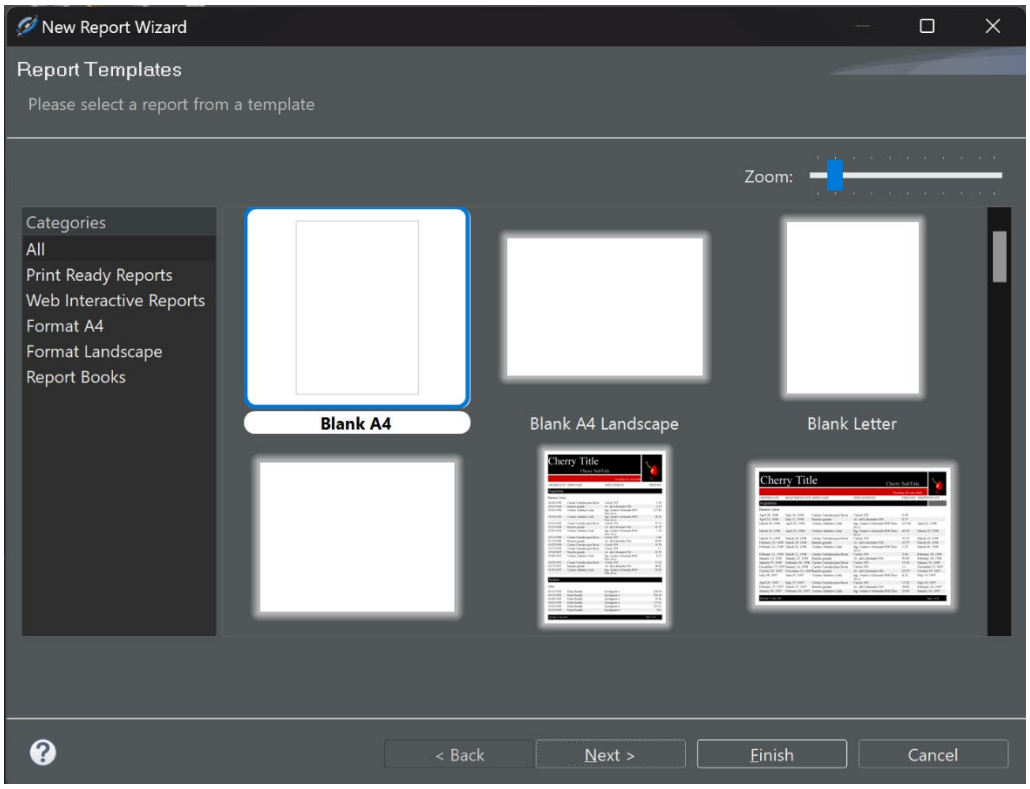

Рисунок 1 – Выбор шаблона

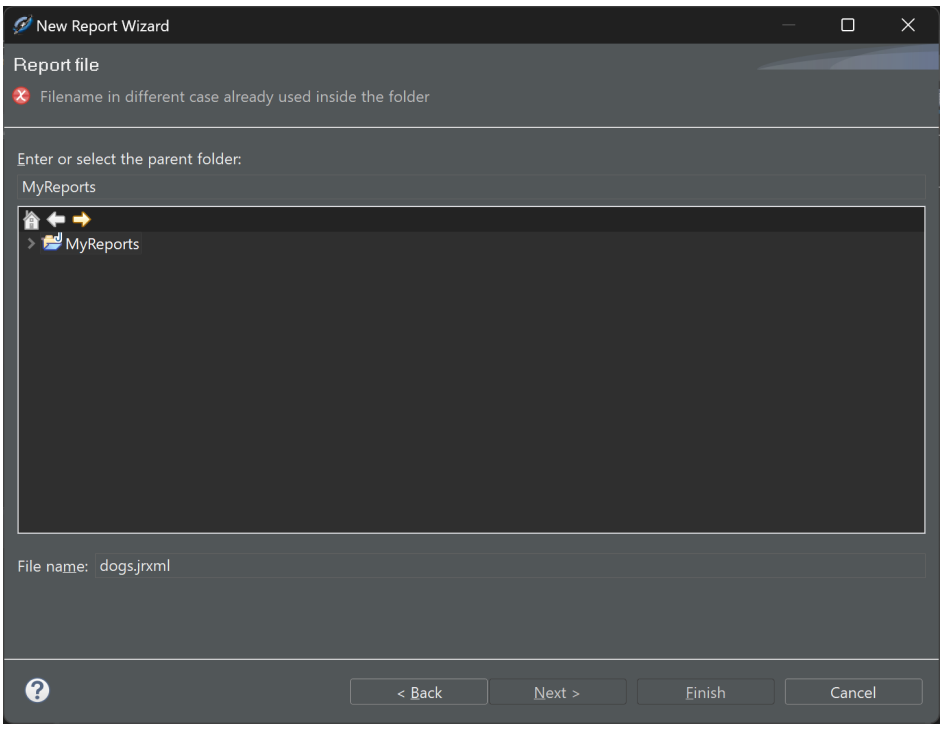

Рисунок 2 – Выбор папки для файла

Открылось диалоговое окно с выбором источника данных (Рис. 3). Введём путь к нашему файлу XML и в графе Select Expression введём «/drivers/driver». Нажмём кнопку Finish.

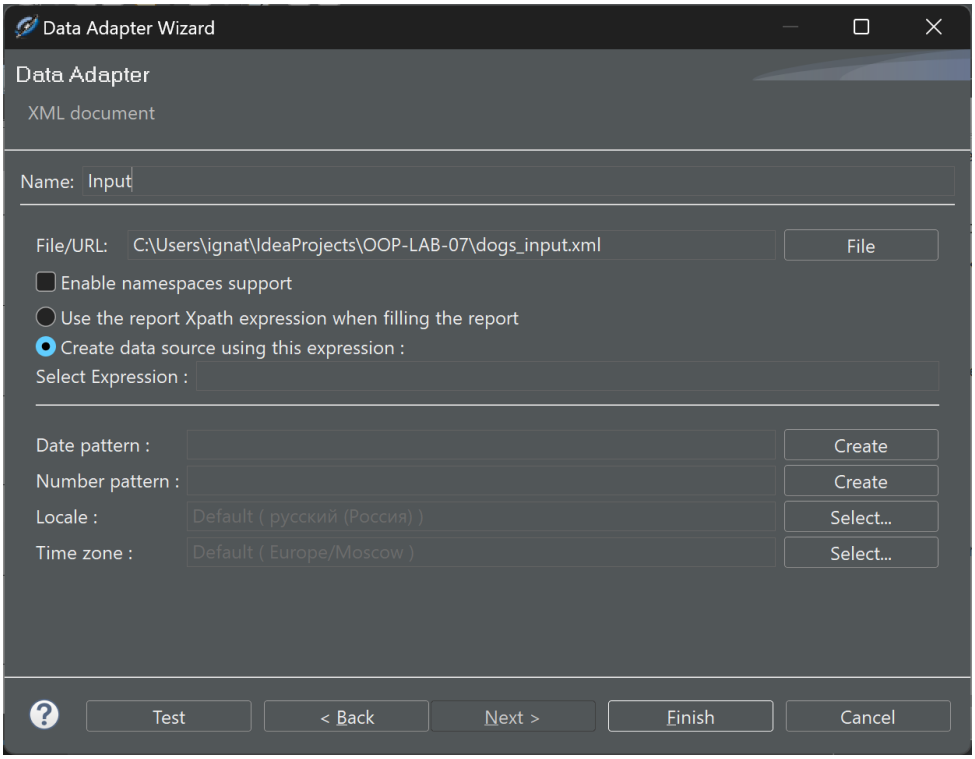

Рисунок 3 – Выбор данных

Теперь перед нами основное окно конструктора (Рис. 4). Здесь мы можем собрать из предложенных элементов скелет нашего отчёта. Создадим подходящий для нашей структуры XML (Рис. 4). Для этого из блока Outline → Fields перетащим в Detail 1 нужные поля. Также украсим наш отчёт элементами из Palette.

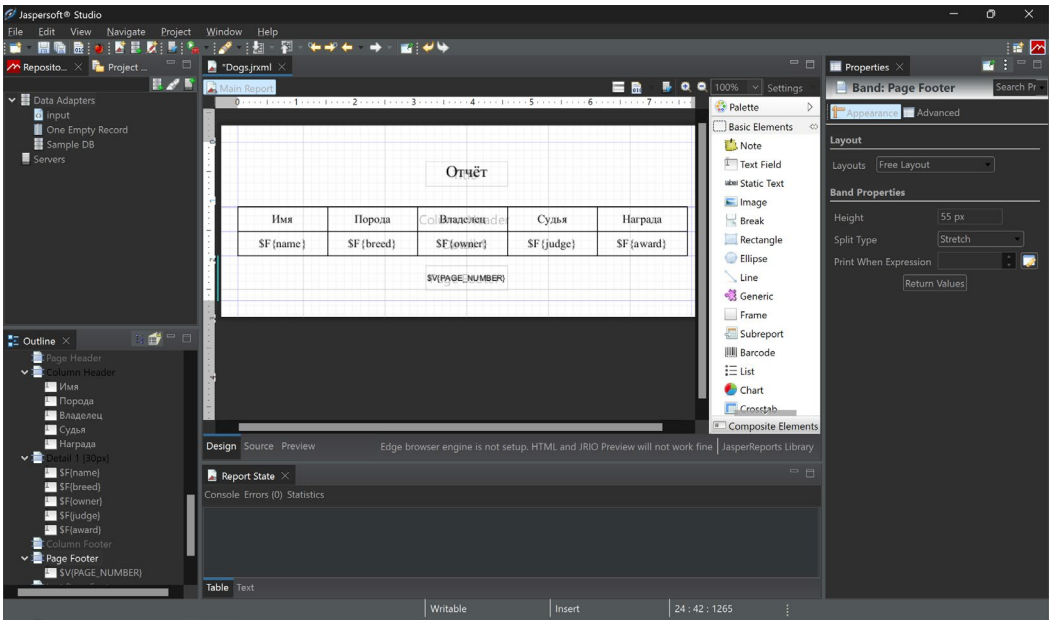

Рисунок 4 – Каркас отчёта

Содержимое файлов

dogs_input.xml

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<doglist>
    <dog name="Рекс" breed="Немецкая овчарка" owner="Иванов И.И." judge="Петров П.П." award="Лучший в породе"/>
    <dog name="Барон" breed="Доберман" owner="Сидорова А.А." judge="Николаев Н.Н." award="Нет наград"/>
    <dog name="Лесси" breed="Колли" owner="Смирнова А.А." judge="Кузнецов К.К." award="Чемпион"/>
    <dog name="Бобик" breed="Бигль" owner="Ковалев В.В." judge="Романов Р.Р." award="Нет наград"/>
    <dog name="Белка" breed="Лабрадор" owner="Новикова Е.В." judge="Филатов Ф.Ф." award="Приз зрительских симпатий"/>
    <dog name="Шерлок" breed="Бассет-хаунд" owner="Орлов О.О." judge="Федоров Н.Н." award="Лучшая стойка"/>
    <dog name="Дина" breed="Ротвейлер" owner="Попова А.А." judge="Сидоров С.С." award="За отвагу"/>
    <dog name="Тайсон" breed="Мопс" owner="Кузнецова Л.С." judge="Михайлов М.М." award="Самый дружелюбный"/>
    <dog name="Арчи" breed="Далматин" owner="Васильева В.В." judge="Павлов П.П." award="Лучшая окраска"/>
    <dog name="Мухтар" breed="Шарпей" owner="Филиппов А.В." judge="Ильин И.И." award="Самый красивый"/>
</doglist>
```

Рисунок 5 – Исходный файл

Отчёт

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

Рисунок 6 – Сгенерированный файл

Текст программы

Класс Main.java

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

/**
 * 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 = new DefaultTableModel(columns, 0);
        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 -> loadDataFromXMLFile());

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

        // 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) {
                        saveDataToXMLFile();
                        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 loadDataFromXMLFile() {
        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 {
                DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
                DocumentBuilder builder = factory.newDocumentBuilder();
                Document doc = builder.parse(file);
                doc.getDocumentElement().normalize();

                tableModel.setRowCount(0); // Очистка данных таблицы перед
загрузкой новых данных

                NodeList dogList = doc.getElementsByTagName("dog");
                for (int i = 0; i < dogList.getLength(); i++) {
                    Node dogNode = dogList.item(i);
                    if (dogNode.getNodeType() == Node.ELEMENT_NODE) {
                        Element dogElement = (Element) dogNode;
                        String name = dogElement.getAttribute("name");
                        String breed = dogElement.getAttribute("breed");
                        String owner = dogElement.getAttribute("owner");
                        String judge = dogElement.getAttribute("judge");
                        String award = dogElement.getAttribute("award");

                        // Добавление строки в таблицу
                        tableModel.addRow(new String[]{name, breed, owner, judge,
award});
                    }
                }
                unsavedChanges = true;
                JOptionPane.showMessageDialog(mainFrame, "Данные успешно загружены
из XML файла");

            } catch (Exception e) {
                JOptionPane.showMessageDialog(mainFrame, "Ошибка при загрузке
```

```
данных из XML файла");
                e.printStackTrace();
            }
        }
    }


    /**
     * Метод для сохранения данных из таблицы в файл.
     */
    private void saveDataToXMLFile() {
        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 {
                DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
                DocumentBuilder builder = factory.newDocumentBuilder();
                Document doc = builder.newDocument();

                // Создание корневого элемента doglist
                Element rootElement = doc.createElement("doglist");
                doc.appendChild(rootElement);

                // Добавление элементов dog в корневой элемент
                for (int i = 0; i < tableModel.getRowCount(); i++) {
                    Element dogElement = doc.createElement("dog");
                    dogElement.setAttribute("name", (String)
tableModel.getValueAt(i, 0));
                    dogElement.setAttribute("breed", (String)
tableModel.getValueAt(i, 1));
                    dogElement.setAttribute("owner", (String)
tableModel.getValueAt(i, 2));
                    dogElement.setAttribute("judge", (String)
tableModel.getValueAt(i, 3));
                    dogElement.setAttribute("award", (String)
tableModel.getValueAt(i, 4));
                    rootElement.appendChild(dogElement);
                }

                // Настройка и выполнение трансформации для записи в файл
                TransformerFactory transformerFactory =
TransformerFactory.newInstance();
                Transformer transformer = transformerFactory.newTransformer();
                transformer.setOutputProperty(OutputKeys.INDENT, "yes");
                DOMSource domSource = new DOMSource(doc);
                StreamResult streamResult = new StreamResult(file);
                transformer.transform(domSource, streamResult);

                unsavedChanges = false;
                JOptionPane.showMessageDialog(mainFrame, "Данные успешно сохранены
в XML файл");

            } catch (Exception e) {
                JOptionPane.showMessageDialog(mainFrame, "Ошибка при сохранении
данных в XML файл");
                e.printStackTrace();
```

```
            }
        }
    }

    /**
     * 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);

            JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 10,
0));
            JButton confirmButton = new JButton("OK");
            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-07