МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ ПО ДИСЦИПЛИНЕ «ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»

Тема: «Создание программного комплекса средствами объектно-ориентированного программирования»

Студент гр. 3312 Лебедев И.А.

Преподаватель Павловский М.Г.

Санкт-Петербург
2024

Введение

Курсовая работа посвящена созданию программного комплекса (ПК) для администратора выставки собак. Данная программа предлагает удобный интерфейс для управления данными о судьях выставки, участниках и их питомцах, лауреатах, а также о месте проведения мероприятия. Основная цель заключается в упрощении работы менеджера, автоматизации процессов ввода данных и получения аналитической информации.

При разработке программного кода на языке Java необходимо учитывать следующие обязательные требования

- Использование закрытых и открытых членов классов.
- Применение наследования.
- Создание конструкторов с параметрами.
- Определение абстрактных базовых классов.
- Реализация виртуальных функций.
- Обработка исключительных ситуаций.
- Динамическое создание объектов.

Техническое задание

Задание 13. Разработать ПК для администратора выставки собак. В ПК должны храниться сведения о собаках, их владельцах и судьях. Администратор выставки может добавлять, изменять и удалять эти сведения. Ему может потребоваться следующая информация:

- список судей и какие породы они обслуживают;
- какая кличка и порода собаки у данного владельца;
- какие собаки получили награды на выставке;
- список владельцев собак указанной породы.

Технические возможности ПК

1. Управление данными о судьях, включая возможность добавления, редактирования и удаления информации.
2. Управление данными о участниках выставки, включая информацию о питомцах и их владельцах.

3. Управление данными о выставках, включая расписание проведения мероприятий и информацию о призерах.
4. Отображение состава участников для каждой выставки.
5. Генерация информации о призерах выставок и их питомцах.
6. Создание и редактирование расписания проведения выставок.
7. Вывод информации о результатах участников, с возможностью сортировки и фильтрации данных.
8. Экспорт данных в формат PDF и HTML.
9. Поддержка работы с файлами XML для сохранения и загрузки данных.
10. Логирование событий для отслеживания работы приложения.

Описание Программного Комплекса

Программный комплекс разработан на языке Java и представляет собой приложение с графическим интерфейсом для управления информацией о собаках и их владельцах. В приложении используются компоненты *JTable* и *DefaultTableModel*, что позволяет эффективно отображать и управлять данными в табличном формате.

Основные функции приложения включают:

- *Работа с таблицами*: Приложение предоставляет возможность отображать и редактировать информацию о судьях, участниках, награжденных и местах проведения выставок. Каждая категория данных представлена на отдельной вкладке, что облегчает навигацию.
- *Добавление и удаление данных*: Пользователи могут добавлять новые записи, удалять выбранные строки и очищать данные в таблицах с помощью интуитивно понятных кнопок.
- *Поиск и фильтрация*: Встроенные инструменты поиска позволяют пользователям находить записи, по ключевым словам, а также сбрасывать результаты поиска для возврата к исходным данным.
- *Экспорт данных*: Приложение поддерживает экспорт данных в форматы PDF и HTML, что позволяет пользователям

генерировать отчеты по выставкам и сохранять их для дальнейшего использования.

- *Регистрация участников*: В приложении предусмотрена функция регистрации владельцев собак, что позволяет вести учет участников выставок.

Интерфейс пользователя включает кнопки для выполнения различных операций, таких как добавление, редактирование и удаление данных, а также инструменты для фильтрации и сортировки записей. Приложение также использует библиотеку *JasperReports* для генерации отчетов, что обеспечивает высокое качество и гибкость в представлении информации.

Проектирование Программного Комплекса

На этапе проектирования приложения для управления списком собак и их владельцев важно детализировать функциональные требования, выделив ключевые процессы, происходящие в данной области. Эти процессы можно описать с помощью прецедентов (use case), которые представляют собой сценарии взаимодействия пользователя с приложением.

Прецеденты:

Каждый прецедент изображается в виде эллипса с названием, содержащим глагол, который отражает суть выполняемой функции.

- *Добавить собаку*: Пользователь может создать новую запись о собаке в списке, загружая данные из XML-файла.
- *Удалить собаку*: Пользователь может удалить выбранную запись о собаке из списка.
- *Сохранить данные*: Пользователь может сохранить изменения в списке собак в XML-файл.
- *Загрузить данные*: Пользователь может загрузить список собак из XML-файла.
- *Поиск собаки*: Пользователь может осуществить поиск собаки, по ключевым словам, таким как имя владельца или кличка собаки.

- *Регистрация владельца*: Новый пользователь может зарегистрироваться в системе для доступа к функционалу приложения.
- *Генерация отчетов*: Пользователь может создать отчеты в формате PDF или HTML для выбранных записей о собаках.

Акторы:

Акторы, взаимодействующие с приложением, могут включать:

- *Пользователь*: Основной актор, который управляет списком собак, добавляет, удаляет и ищет записи.
- *Администратор*: Актор, который отвечает за управление пользователями и их правами доступа.

Связи между актерами и прецедентами

Каждый из акторов взаимодействует с прецедентами, выполняя определенные действия в приложении, что позволяет эффективно управлять данными о собаках и их владельцах.

На диаграмме прецедентов связи между актерами и прецедентами отображаются стрелками, указывающими на инициатора взаимодействия. Например, стрелка от актора "Пользователь" к прецеденту "Добавить собаку" демонстрирует, что пользователь инициирует этот процесс.

- *Связи использования и расширения*
- *Связь использования (uses)*: если один прецедент всегда использует функциональность другого, это можно отразить с помощью связи использования. Например, прецеденты "Сохранить данные" и "Загрузить данные" могут использовать один и тот же механизм для работы с файлами.
- *Связь расширения (extends)*: если один прецедент может быть расширен другим, это также следует отразить. Например, прецедент "Поиск собаки" может быть расширен прецедентом "Фильтрация по критериям", если пользователь хочет уточнить поиск.
- *Ранжирование прецедентов*: Прецеденты необходимо упорядочить по приоритету, чтобы в начальных циклах разработки

реализовать наиболее важные функции. Например, прецеденты "Добавить собаку" и "Удалить собаку" могут быть приоритетными, так как они составляют основную функциональность приложения.

Элементы диаграммы прецедентов

Акторы:

- *Пользователь*: Основной актор, который взаимодействует с приложением.
- *Администратор*: Актор, который управляет пользователями и их правами доступа.

Прецеденты:

- *Добавить собаку*: Пользователь может внести новую запись о собаке в систему.
- *Удалить собаку*: Пользователь может удалить запись о собаке из списка.
- *Сохранить данные*: Пользователь может сохранить изменения в списке собак.
- *Загрузить данные*: Пользователь может загрузить список собак из файла.
- *Поиск собаки*: Пользователь может искать собаку по ключевым словам.
- *Регистрация владельца*: Новый пользователь может зарегистрироваться в системе.
- *Генерация отчетов*: Пользователь может создавать отчеты в формате PDF или HTML.

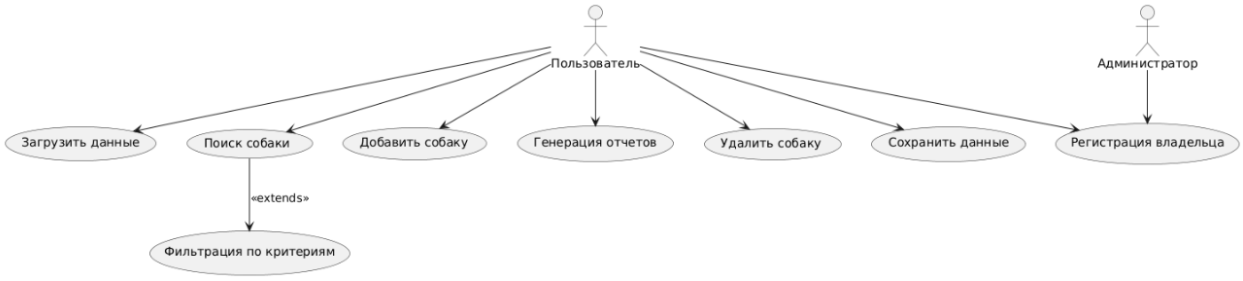
Связи

Связи между актерами и прецедентами изображаются стрелками, указывающими на инициатора взаимодействия.

- Пользователь будет связан со всеми прецедентами, так как он инициирует все действия.

- Администратор может быть связан с прецедентами, касающимися управления пользователями (например, регистрация владельца).

Диаграмма прецедентов



Создание прототипа пользовательского интерфейса

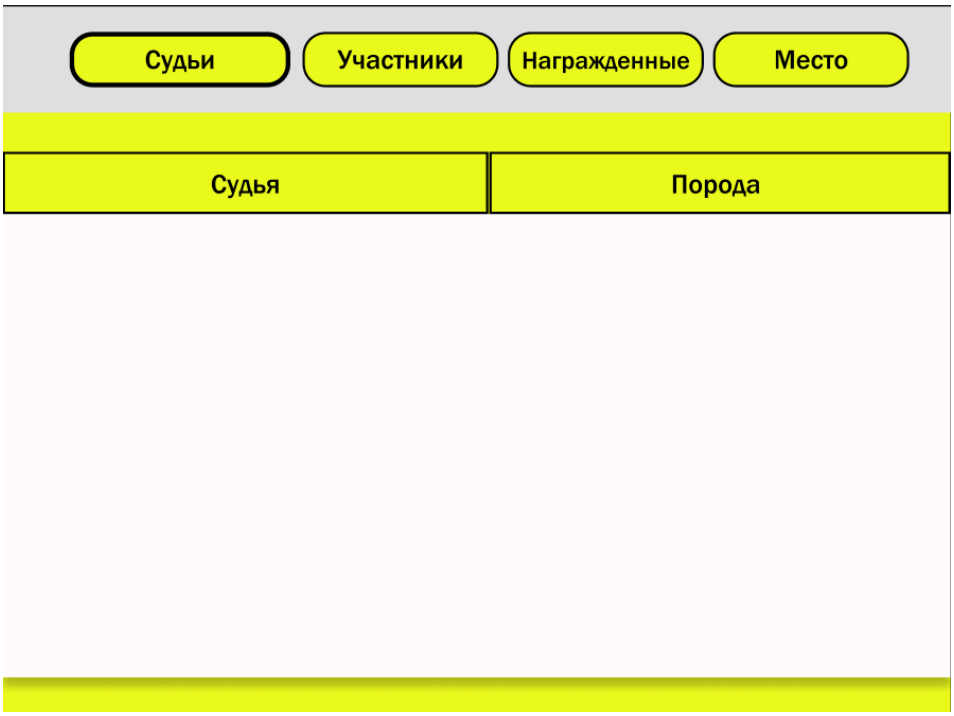
Описание прецедента представляет собой общее представление процесса, не углубляясь в детали его реализации. Важно отметить, что проектные решения, касающиеся пользовательского интерфейса, остаются вне рамок этого обсуждения. Для разработки интерфейса пользователя необходимо рассмотреть процесс с учетом реальных проектных решений, основанных на конкретных технологиях ввода и вывода информации.

Когда речь заходит о пользовательском интерфейсе, прецеденты разбиваются на экранные формы, которые определяют содержимое диалоговых окон и описывают способы взаимодействия с различными устройствами. Каждая экранная форма должна включать в себя поля ввода, а также перечень элементов управления. Важно также описать действия пользователя, такие как нажатие кнопки, выбор пункта меню, ввод данных или использование правой и левой кнопок мыши, а также отклики системы, например, отображение данных, вывод подсказок или перемещение курсора.

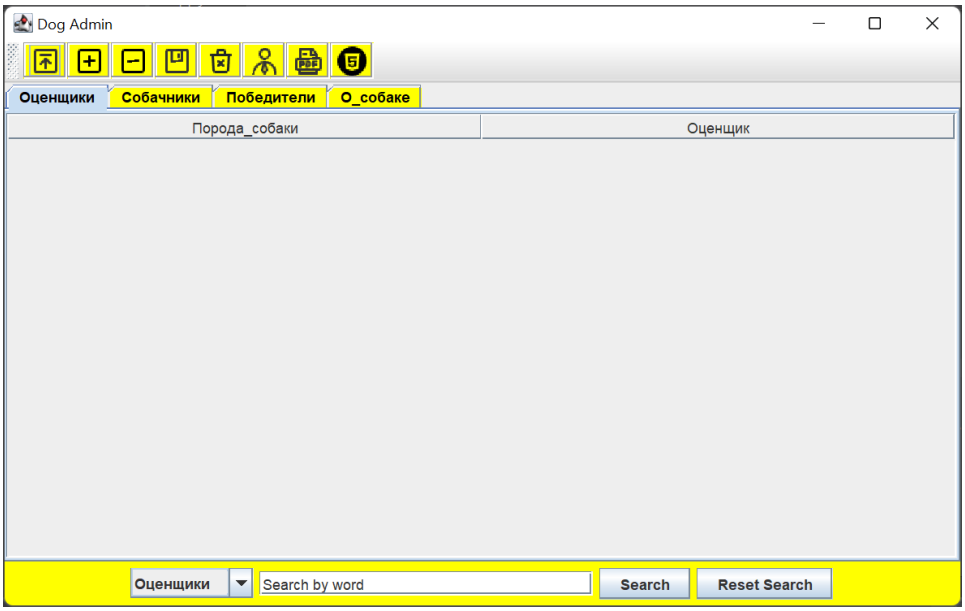
Создание прототипа пользовательского интерфейса — это процесс, который позволяет визуализировать взаимодействие пользователя с приложением. Он включает в себя не только проектирование экранных форм, но и продумывание логики взаимодействия, чтобы обеспечить интуитивно понятный и удобный опыт для пользователя. В результате, хорошо спроектированный интерфейс не только улучшает восприятие приложения,

но и способствует более эффективному выполнению задач, что делает его важным аспектом разработки программного обеспечения.

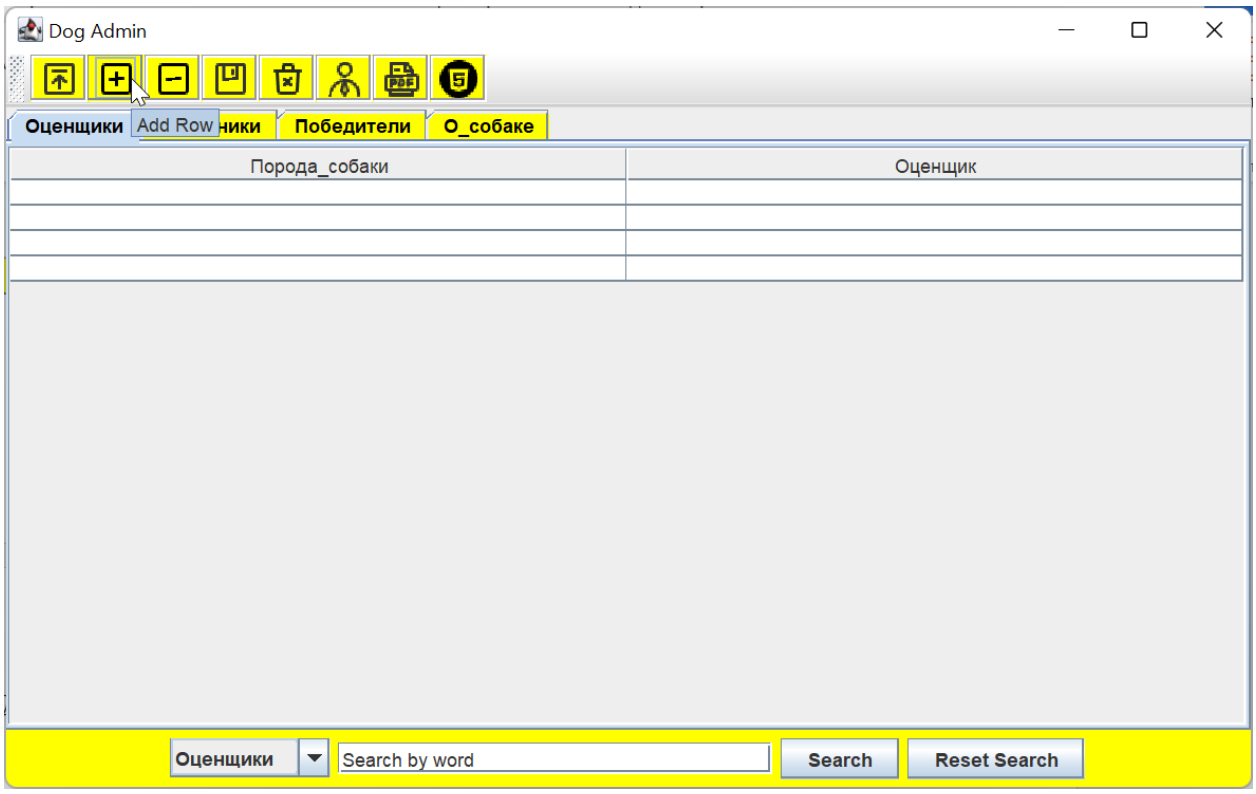
Создание прототипа интерфейса пользователя



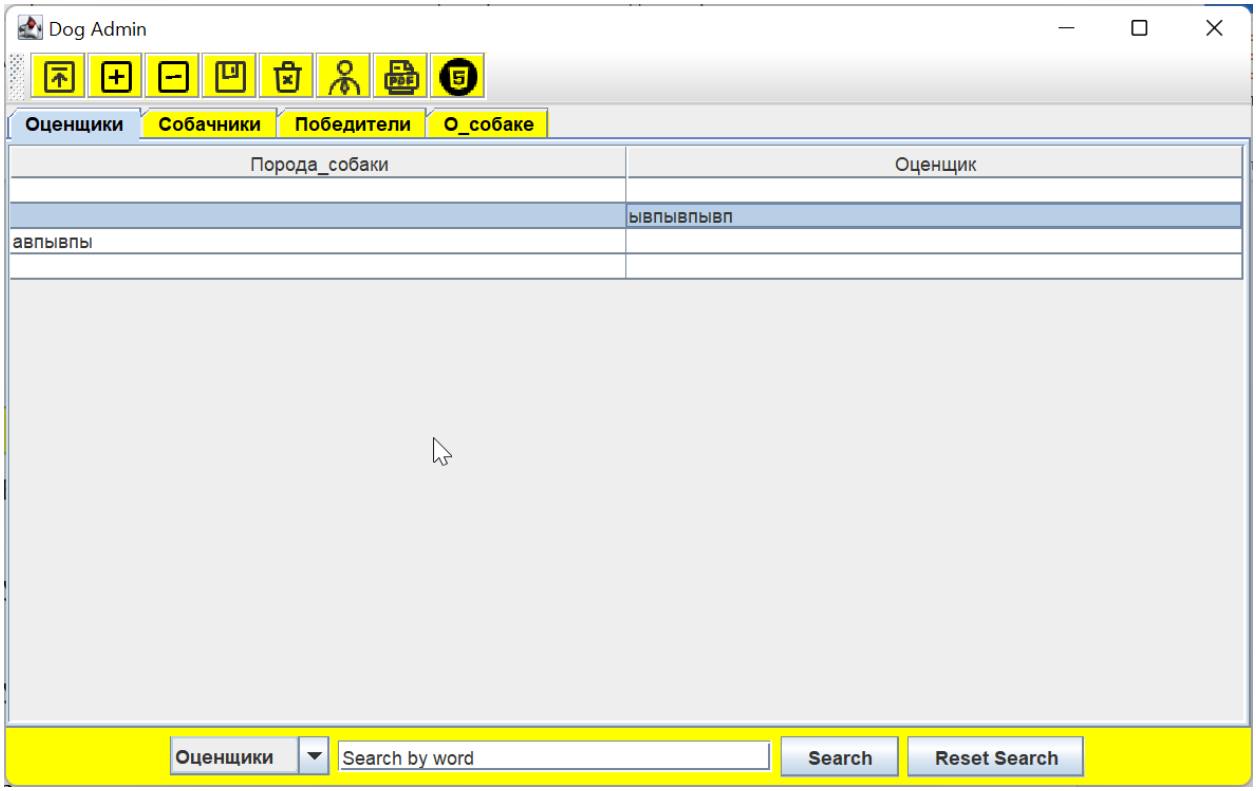
Главное окно будет иметь следующий вид



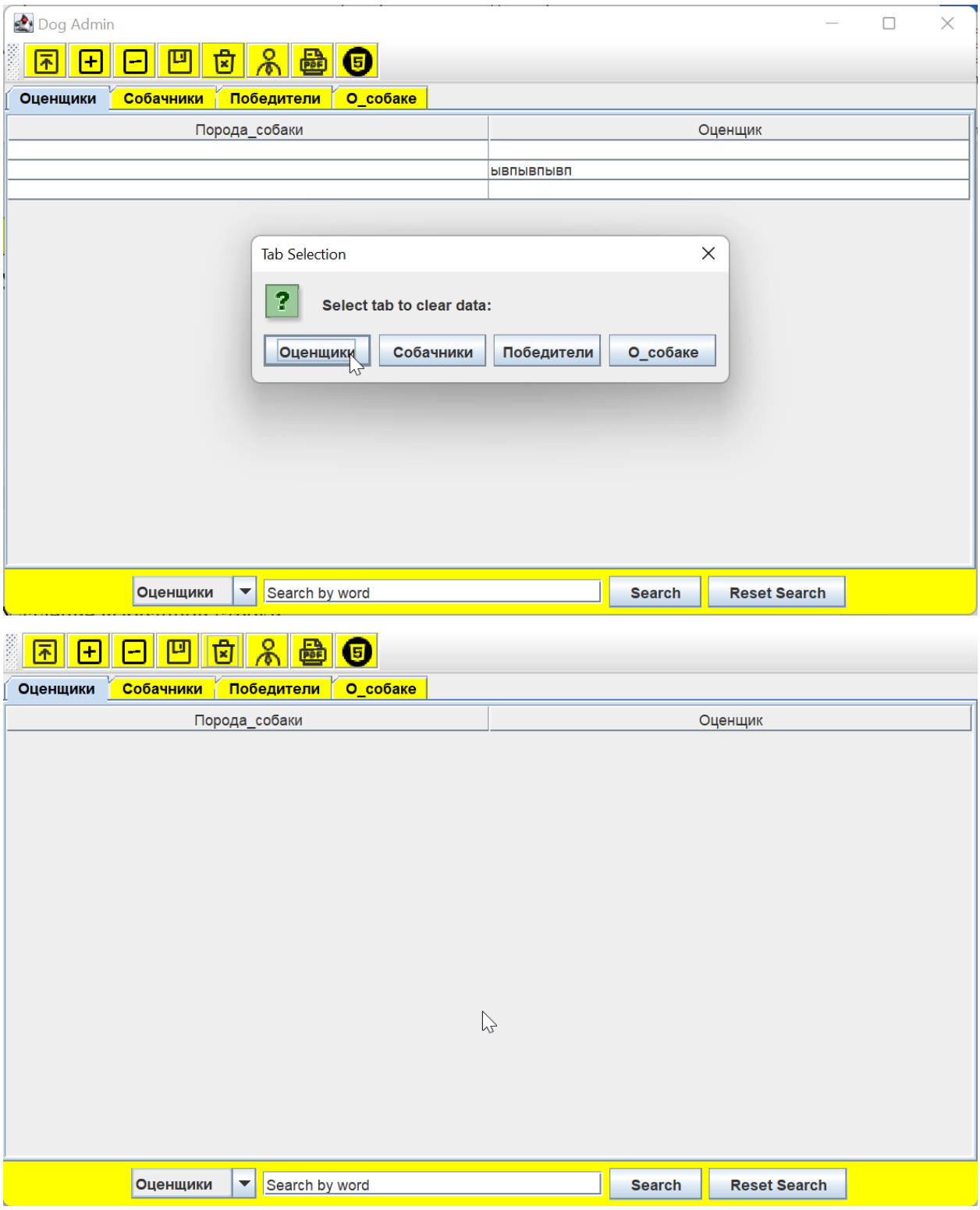
Готовый результат



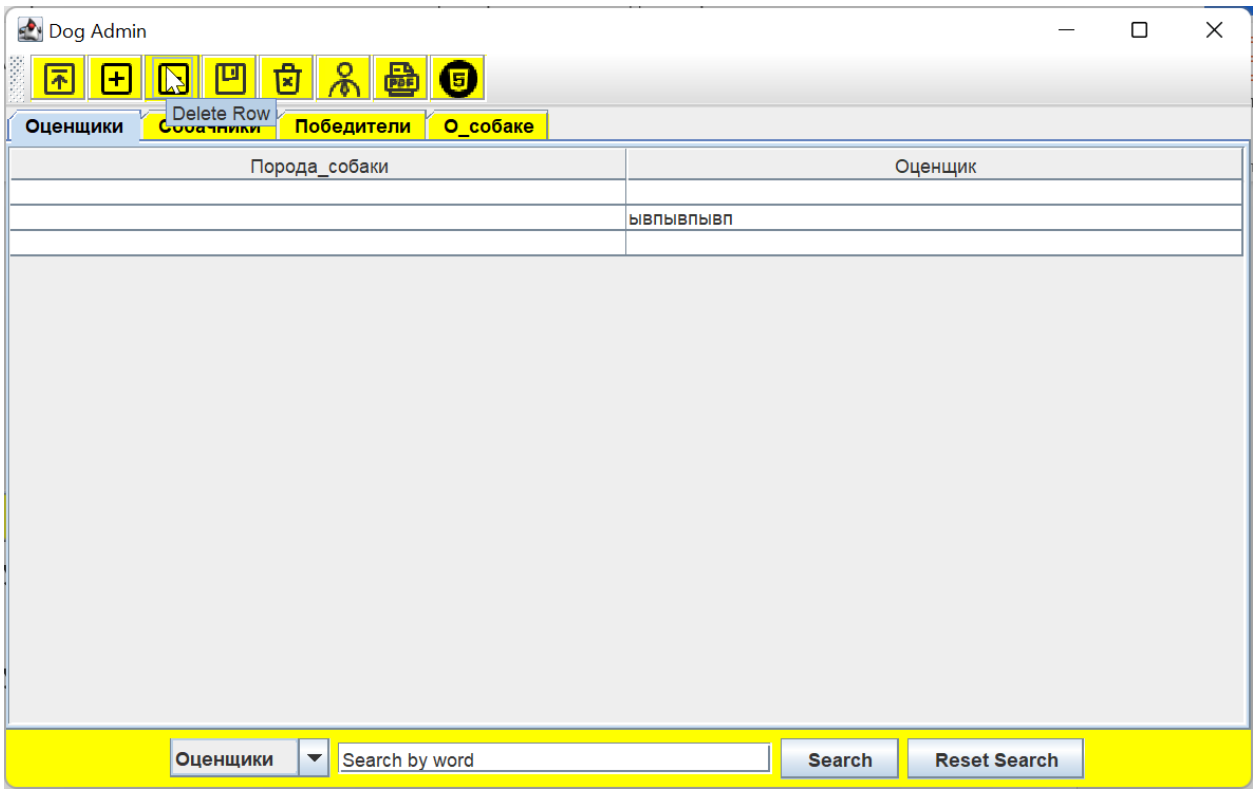
Добавление пустой строки



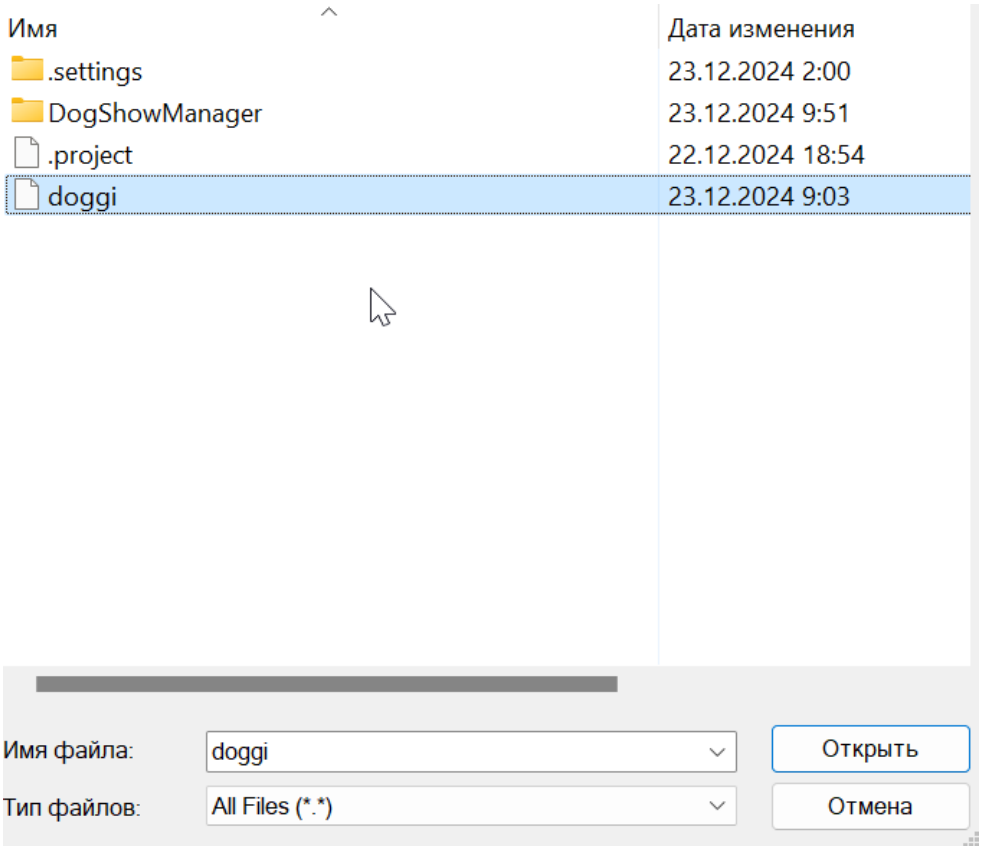
Возможность редактирования

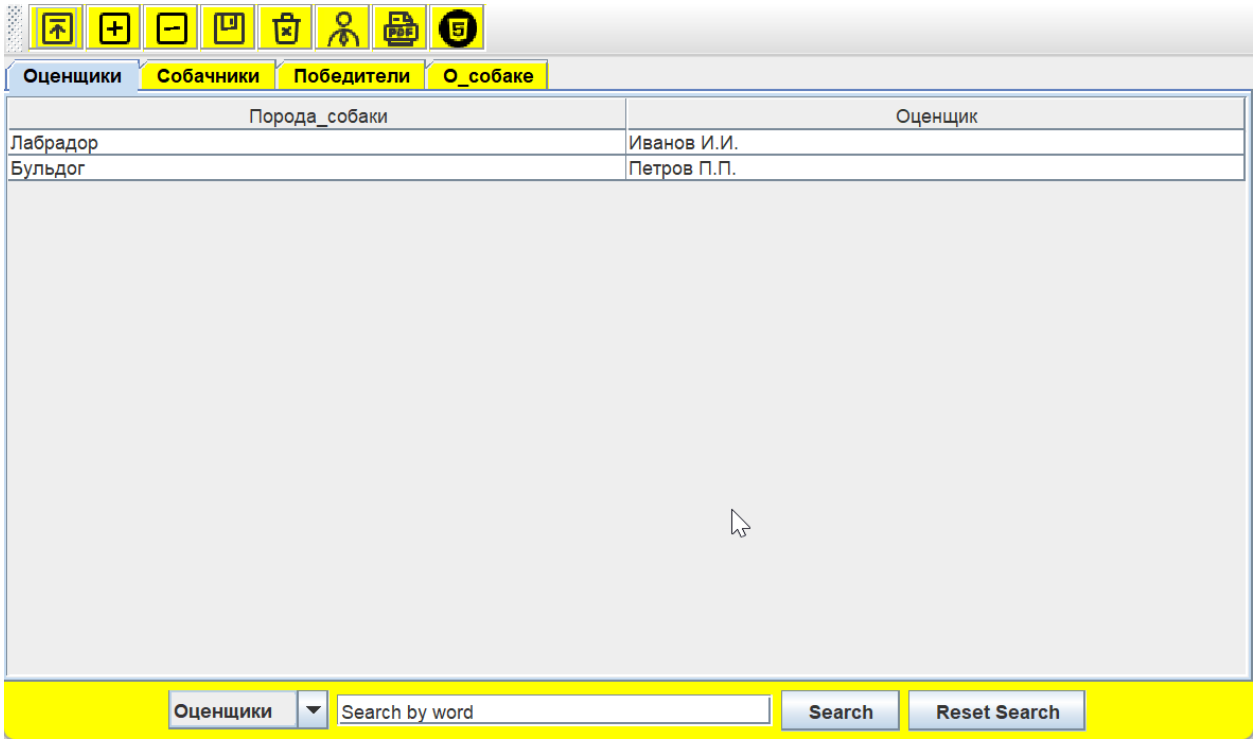


Удаление выбранной вкладки

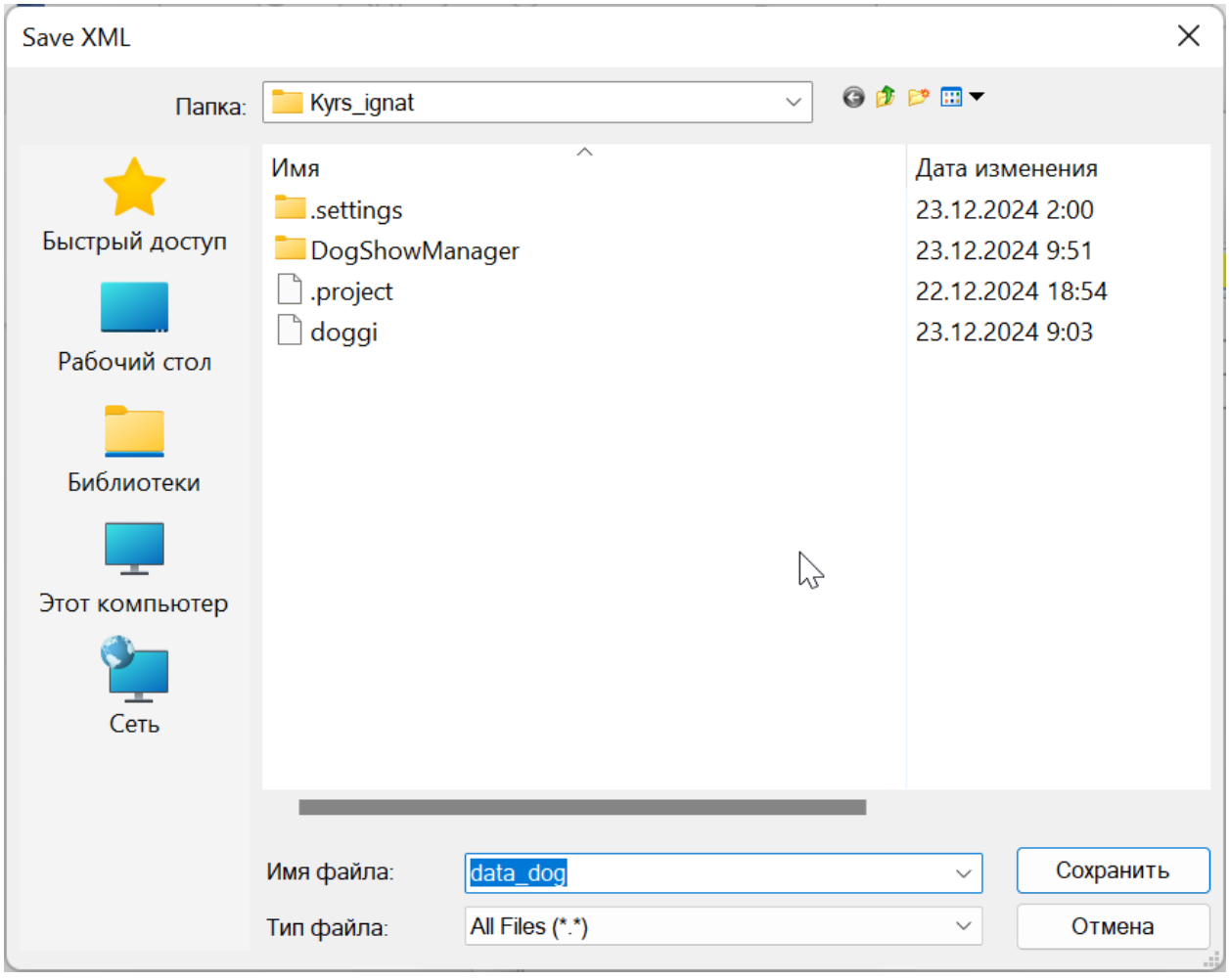


Удаление выбранной строки

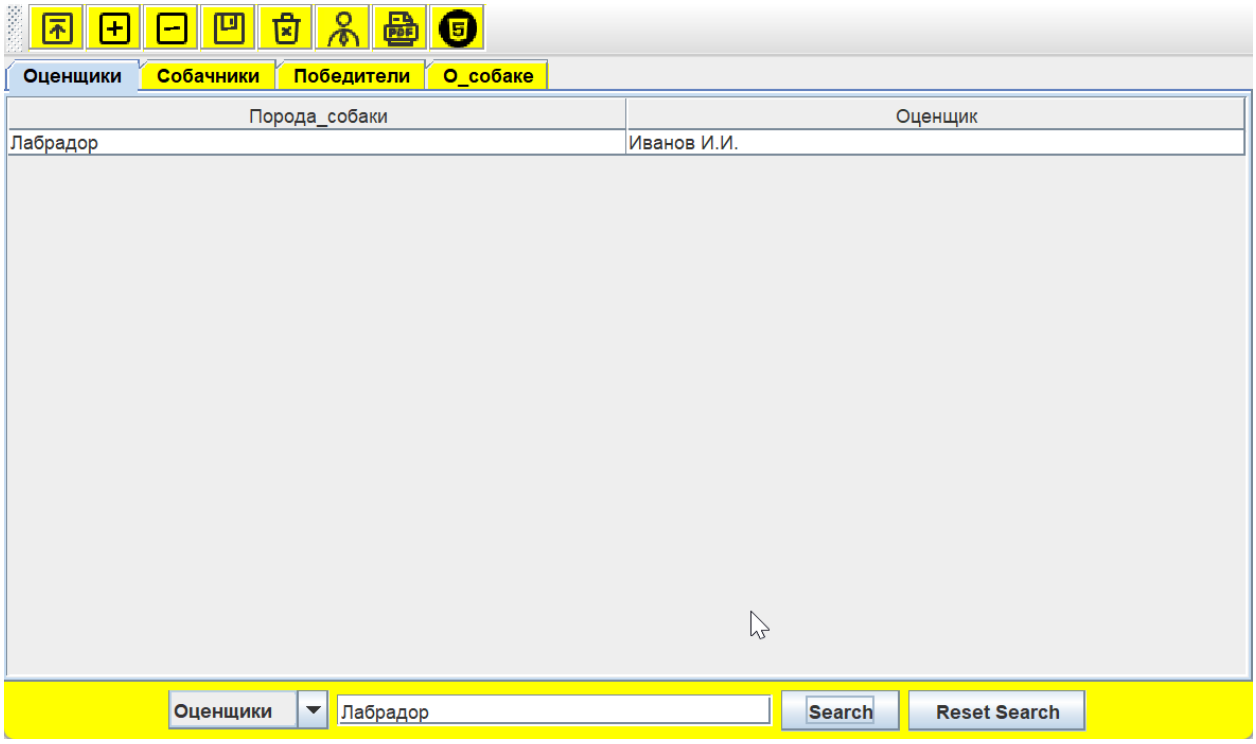


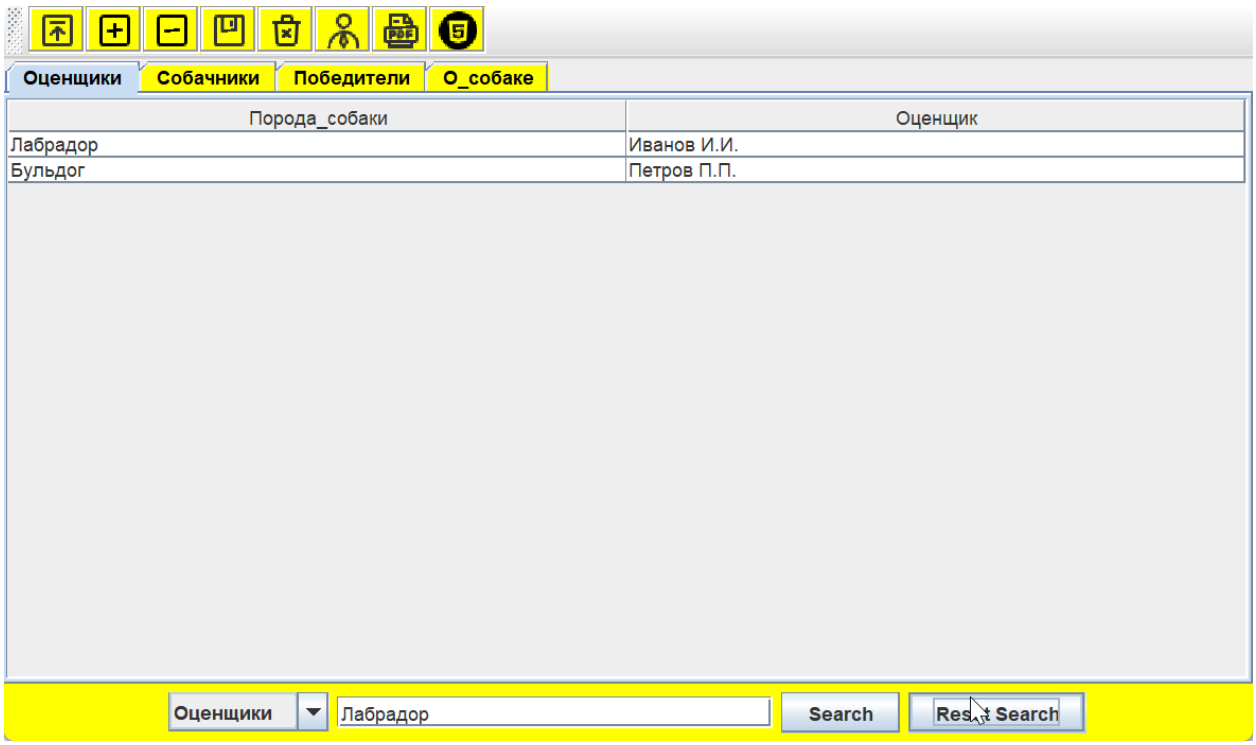


Загрузка данных

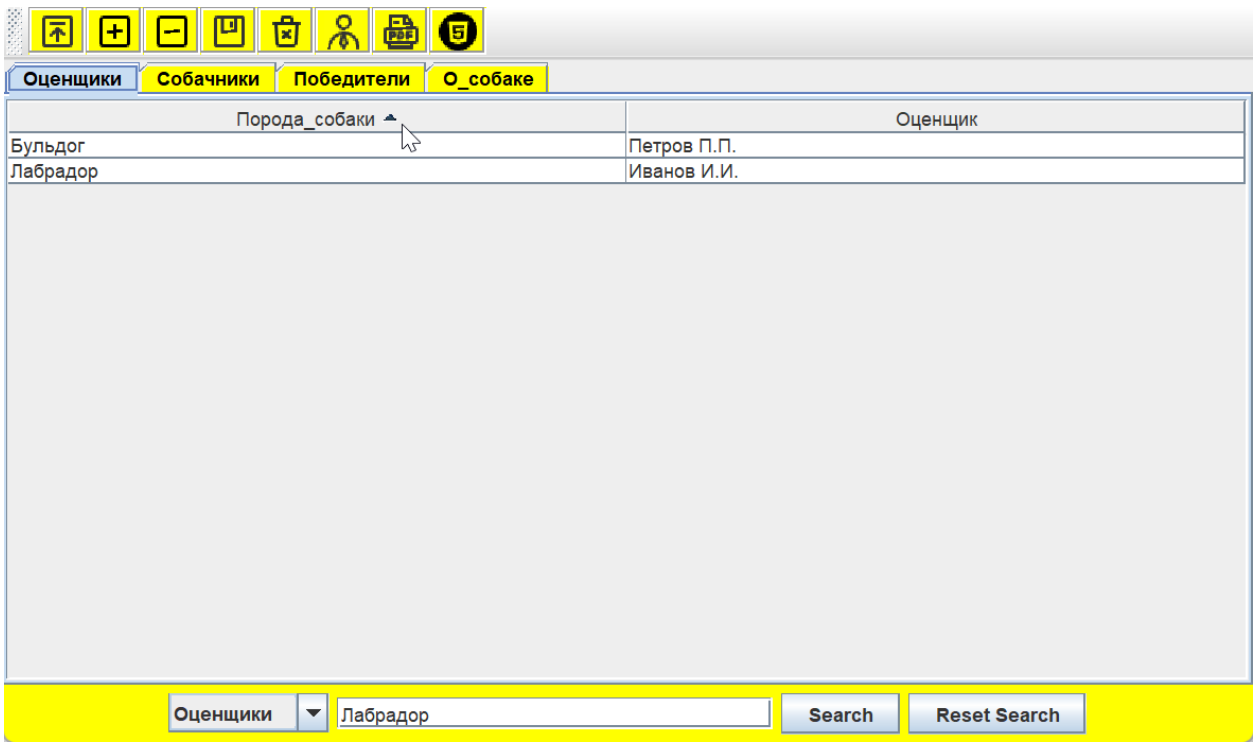


Сохранение данных

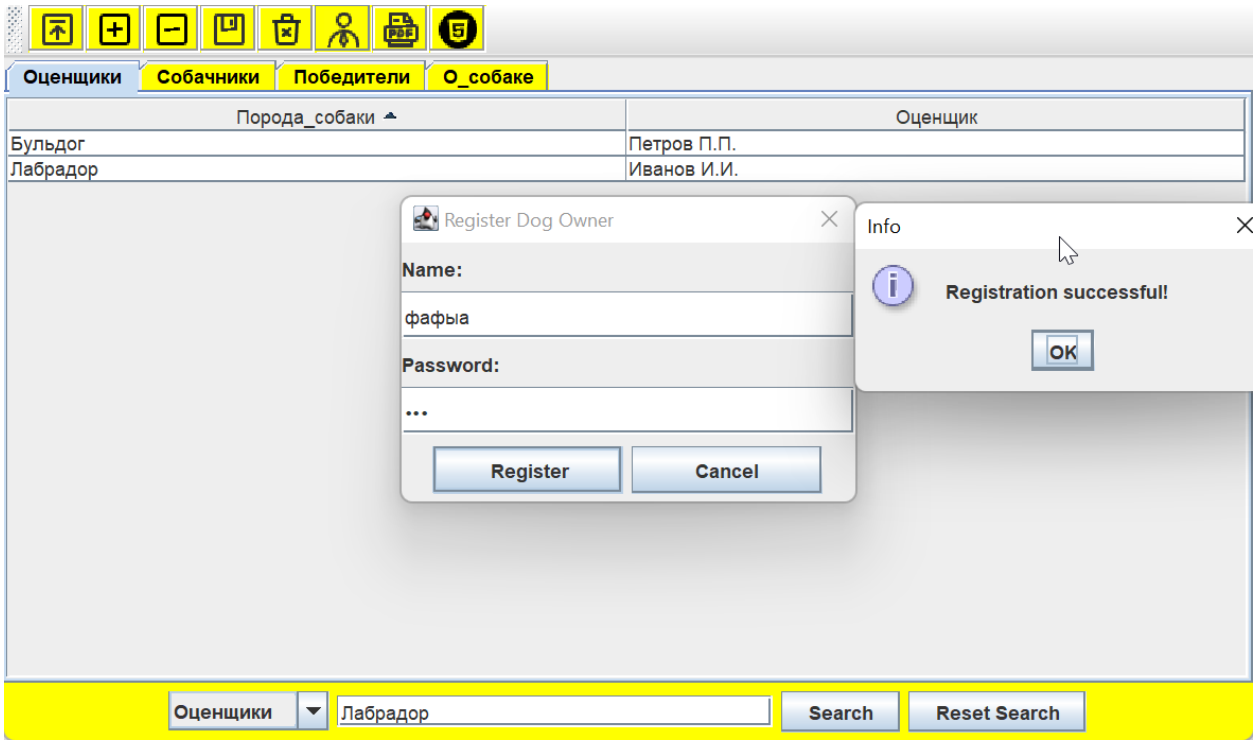




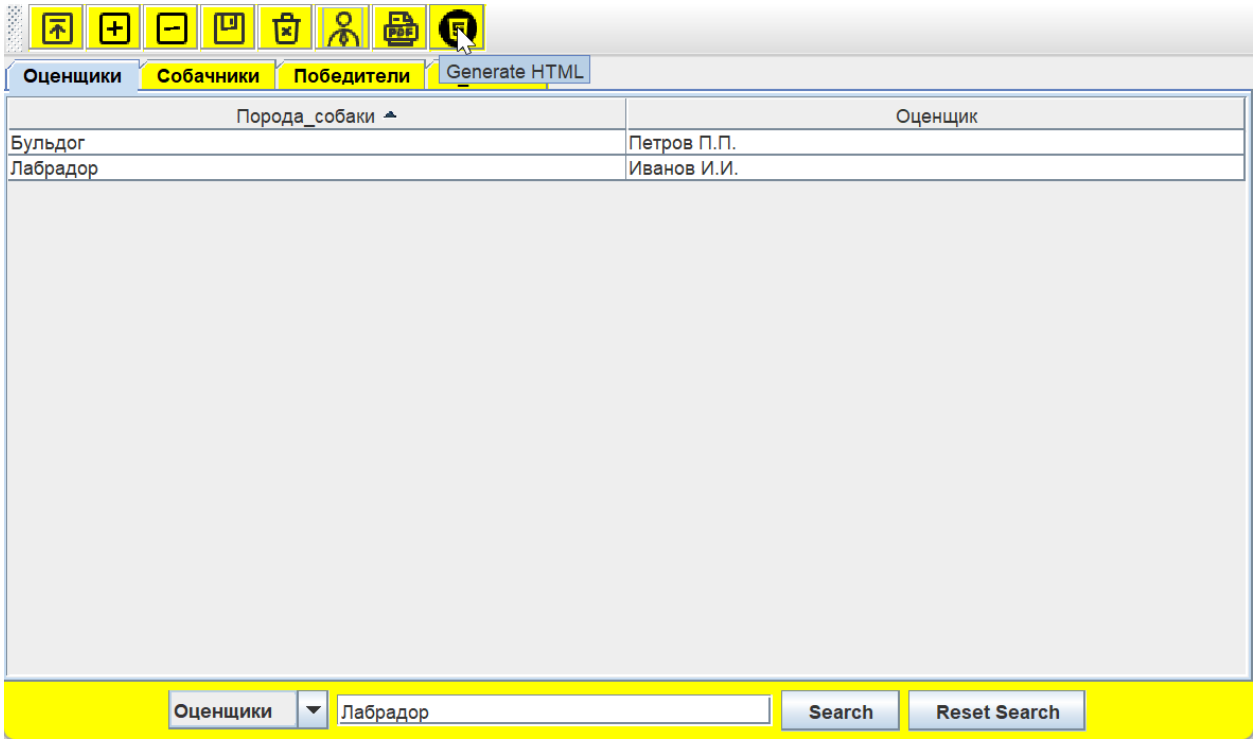
Поиск по ключевому слову

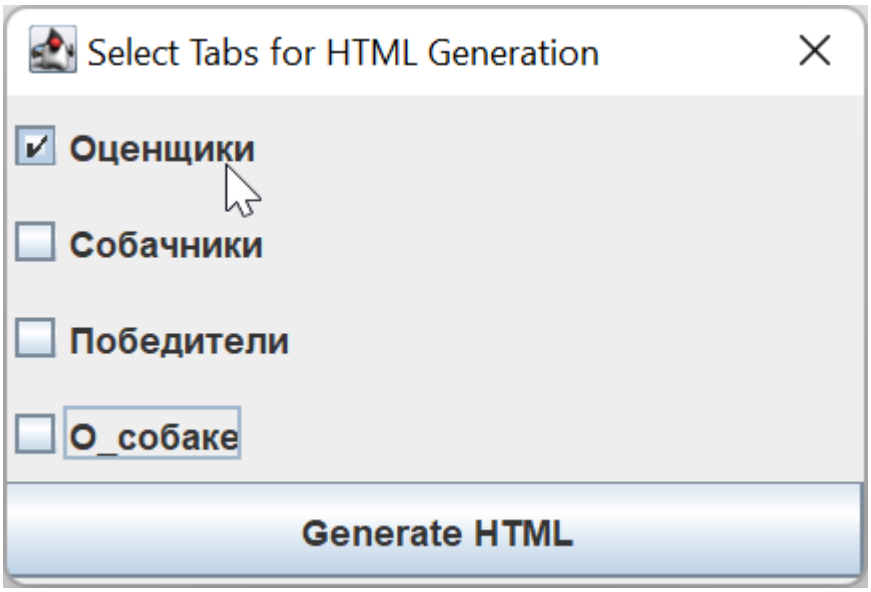


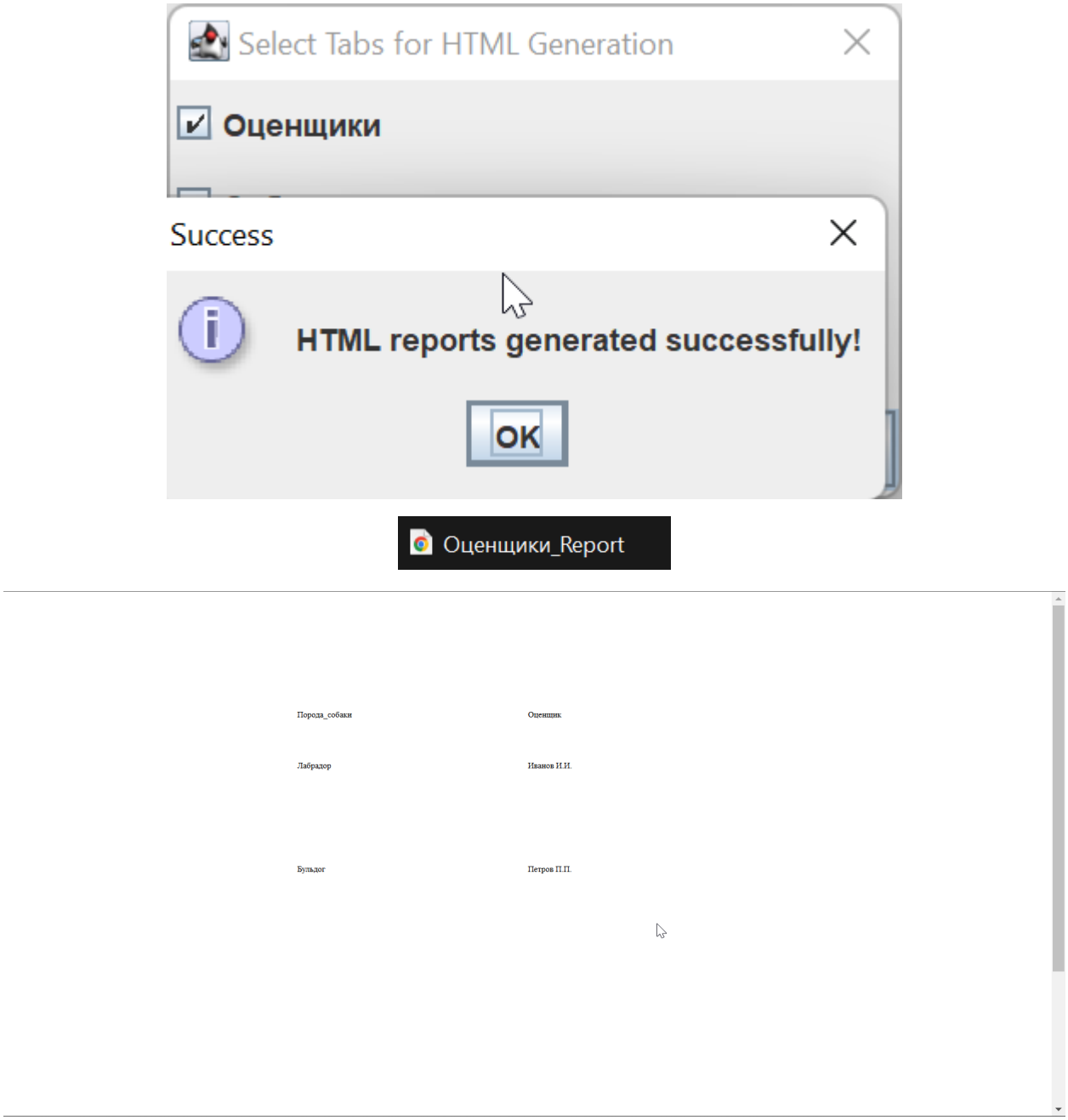
Сортировка



Регистрация пользователя


Оценщики
Собачники
Победители
Generate HTML
Порода_собаки
Оценщик
Бульдог
Петров П.П.
Лабрадор
Иванов И.И.
Оценщики
Лабрадор
Search
Reset Search


Select Tabs for HTML Generation
Оценщики
Собачники
Победители
O_собаке
Generate HTML



Создание HTML отчета выбранной вкладки

Разработка объектной модели приложения для управления списком собак

Объектная модель приложения не просто описывает структуру программного комплекса, а визуализирует ключевые концепции предметной области в виде набора типов объектов (сущностей). Эти сущности формируются путем выделения их из предметной области и анализа прецедентов, связанных с управлением данными о собаках, их владельцах, выставках и судьях.

Сущности

На диаграмме сущность представляется в виде прямоугольника, внутри которого указано имя сущности, ее атрибуты и операции. В нашем приложении можно выделить следующие сущности:

1. *Собака (Dog)*
    - *Атрибуты*:
        - *Имя*: *String* — имя собаки.
        - *Порода*: *String* — порода собаки.
        - *Владелец*: *String* — имя владельца собаки.
    - *Операции*:
        - *добавить()*: добавляет новую собаку в систему.
        - *удалить()*: удаляет собаку из списка.
        - *редактировать()*: обновляет информацию о собаке.
2. *Владелец (Owner)*
    - *Атрибуты*:
        - *Имя*: *String* — имя владельца.
        - *Дата*: *String* — Дата регистрации собаки.
    - *Операции*:
        - *добавить()*: добавляет нового владельца.
        - *удалить()*: удаляет владельца из списка.
        - *редактировать()*: обновляет информацию о перерегистрации.
3. *Судья (Judge)*
    - *Атрибуты*:
        - *Имя*: *String* — имя судьи.
        - *Опыт*: *String* —порода с которой работает.
    - *Операции*:
        - *добавить()*: добавляет нового судью.
        - *удалить()*: удаляет судью из списка.

- *редактировать()*: обновляет информацию о работе с породой.

4. *О собаке (Dog)*
    - *Атрибуты*:
        - *кличка*: *String* — кличка присвоенная собаке.
        - *порода*: *String* — порода собаки
    - *Операции*:
        - *добавить()*: добавляет новую собаку.
        - *удалить()*: удаляет собаку из списка.
        - *редактировать()*: обновляет информацию о собаке.
5. *Хозяин (User)*
    - *Атрибуты*:
        - *Имя*: *String* — имя хозяина.
        - *Пароль*: *String* — пароль для доступа к системе.
    - *Операции*:
        - *регистрация()*: регистрирует нового пользователя.
        - *вход()*: позволяет пользователю войти в систему.
        - *выход()*: завершает сессию пользователя.

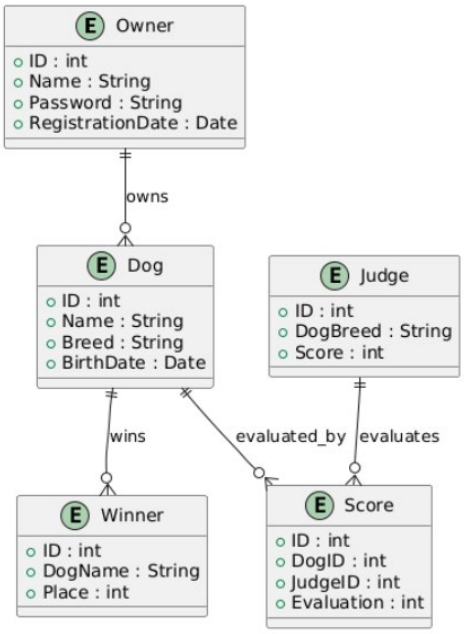
6. *Ассоциации между сущностями*

Ассоциации между сущностями отражают бинарные отношения между ними и обозначаются линиями, соединяющими сущности, с указанием их семантического смысла. В нашем приложении можно выделить следующие ассоциации:

- *Собака и Владелец:*
    - *Ассоциация*: "принадлежит".
    - *Кратность*: 1 .. 1 (каждая собака принадлежит одному владельцу, и каждый владелец может иметь одну или несколько собак).
- *Выставка и Оценивающие*:
    - *Ассоциация*: "судит".

    - *Кратность*: 1 .. * (каждая выставка может быть судима несколькими судьями, и каждый судья может судить несколько выставок).
- *Собака и Выставка*:
    - *Ассоциация*: "участвует в".
    - *Кратность*: 0 .. * (собака может участвовать в нулевом или большом количестве выставок, и каждая выставка может включать несколько собак).

Диаграмма сущностей



Структура классов приложения "Dog Show Manager"

В этом приложении, предназначенном для управления данными о собаках, их владельцах и оценщиках, мы определяем несколько ключевых классов. Каждый класс имеет свои атрибуты и методы, которые позволяют эффективно управлять данными и взаимодействовать с пользователем.

Класс Dog (Собака):

- Атрибуты:
  - *name: String* — имя собаки.
  - *breed: String* — порода собаки.
  - *owner: Owner* — объект класса *Owner*, представляющий владельца собаки.
  - *birthDate: Date* — дата рождения собаки.
- Методы:
  - *add(): boolean* — добавляет новую собаку в систему.
  - *remove(): boolean* — удаляет собаку из системы.
  - *edit(): boolean* — редактирует информацию о собаке.

Класс Owner (Владелец):

- Атрибуты:

- *name: String* — имя владельца.
- *contact: String* — контактная информация владельца.
- *dogs: List<Dog>* — список собак, принадлежащих владельцу.

- Методы:
  - *add(): boolean* — добавляет нового владельца в систему.
  - *remove(): boolean* — удаляет владельца из системы.
  - *edit(): boolean* — редактирует информацию о владельце.

Класс Judge (Судья):

- Атрибуты:
  - *name: String* — имя судьи.
  - *experience: int* — количество лет опыта судейства.
  - *evaluations: List<Evaluation>* — список оценок, выставленных судьей.
- Методы:
  - *add(): boolean* — добавляет нового судью в систему.
  - *remove(): boolean* — удаляет судью из системы.
  - *edit(): boolean* — редактирует информацию о судье.

Класс Exhibition (Выставка):

- Атрибуты:
  - *date: Date* — дата проведения выставки.
  - *location: String* — место проведения выставки.
  - *dogs: List<Dog>* — список собак, участвующих в выставке.
- Методы:
  - *add(): boolea***n** — добавляет новую выставку в систему.
  - *remove(): boolean* — удаляет выставку из системы.
  - *edit(): boolean* — редактирует информацию о выставке.

Класс User (Пользователь):

- Атрибуты:
  - *username: String* — имя пользователя.
  - *password: String* — пароль для доступа к системе.

- Методы:
    - *register(): boolean* — регистрирует нового пользователя в системе.
    - *login(): boolean* — позволяет пользователю войти в систему.
    - *logout(): boolean* — завершает сессию пользователя.

Ассоциации

- Собака и Владелец:
    - Связь "принадлежит" (1 .. 1). Каждая собака имеет одного владельца, и каждый владелец может иметь только одну собаку. Это означает, что каждый объект класса *Dog* связан с одним объектом класса *Owner*.
- Выставка и Судья:
    - Связь "судит" (1 .. *). Каждая выставка может иметь нескольких судей, которые оценивают участников. Это означает, что один объект класса *Exhibition* может быть связан с несколькими объектами класса *Judge*.
- Собака и Выставка:
    - Связь "участвует в" (0 .. *). Одна собака может участвовать в нескольких выставках, а выставка может включать множество собак. Это создает гибкость в управлении участниками выставок.

Агрегация

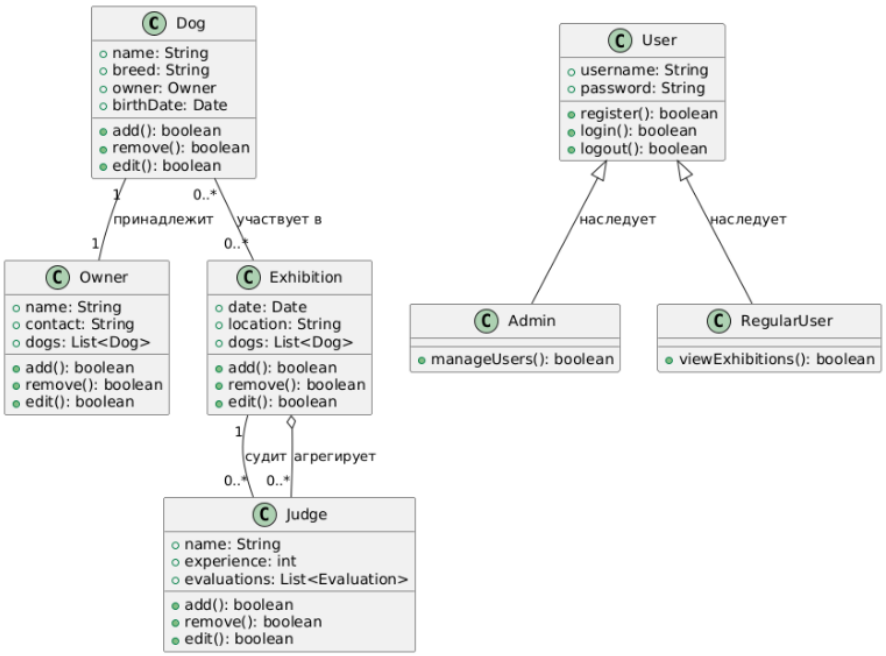
- Выставка и Судья:
    - Выставка агрегирует судей, что означает, что выставка может включать несколько судей, но судьи могут существовать независимо от выставок. Эта связь изображается полым ромбом на линии ассоциации, указывая на то, что *Exhibition* может содержать множество объектов класса *Judge*.

Наследование

- Пользователь:

- Класс *User* может быть базовым классом для производных классов, таких как *Admin* (Администратор) и *RegularUser* (Обычный пользователь). Это означает, что оба класса наследуют общие атрибуты и методы от класса *User* , но могут также иметь свои уникальные характеристики. Эта связь изображается стрелкой с полым треугольником, указывающей от производного класса к базовому.

Диаграмма классов



Описание поведения приложения

Приложение предназначено для управления данными о собаках, их владельцах, судьях и выставках. Оно позволяет пользователям выполнять различные операции, такие как добавление, редактирование и удаление информации о собаках и их владельцах, а также управление выставками и оценками.

1. Идентификация пользователей и объектов

В нашем приложении выделяются два типа пользователей:

- *Администратор*: управляет пользователями и имеет доступ ко всем функциям приложения.

- *Обычный пользователь*: может добавлять и редактировать информацию о своих собаках и участвовать в выставках.

Основные объекты приложения:

- *Собака*
- *Владелец*
- *Оценивающий*
- *Выставка*
- *Пользователь*

2. Выбор операций

В приложении реализованы следующие операции:

- *Добавление*: добавление новой собаки, владельца, судьи или выставки.
- *Редактирование*: редактирование информации о существующих собаках, владельцах, судьях или выставках.
- *Удаление*: удаление собаки, владельца, судьи или выставки.
- *Регистрация и вход*: регистрация нового пользователя и вход в систему.

3. Отображение запросов на выполнение операций

На диаграмме последовательностей каждый запрос на выполнение операции будет представлен горизонтальной линией со стрелкой, указывающей от пользователя к объекту, который выполняет операцию.

4. Порядок выполнения операций

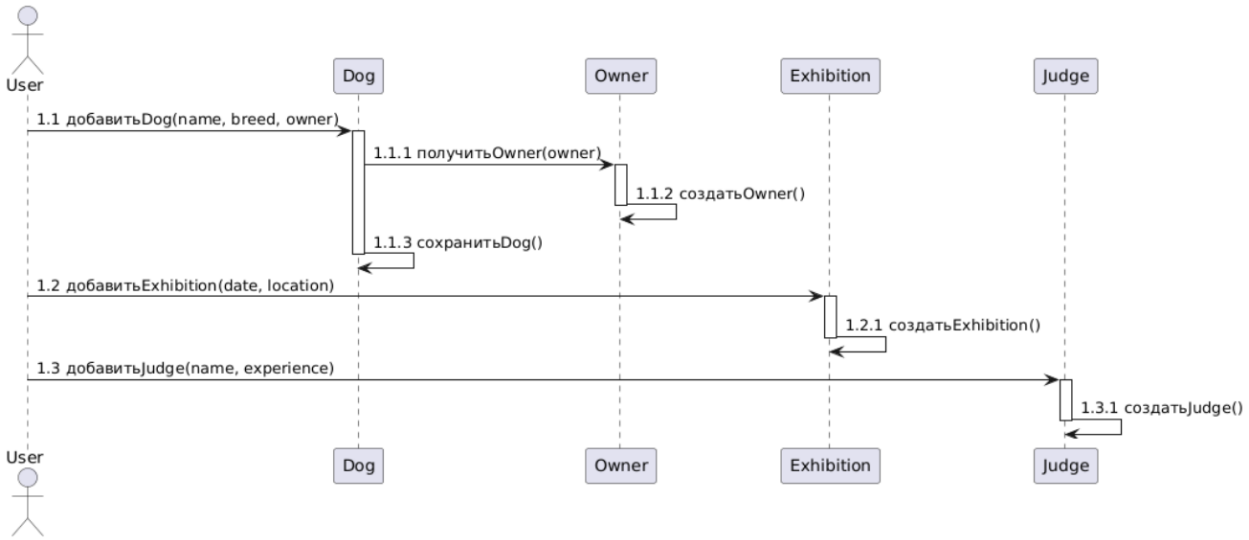
Порядок выполнения операций будет определяться их номером. Чем ниже горизонтальная линия, тем позже выполняется операция. Мы будем использовать вложенную систему нумерации для отображения вложенности операций.

5. Условия и создание объектов

На диаграмме последовательностей можно также отобразить условия выполнения операций и моменты создания и уничтожения объектов.

Например, если пользователь решает удалить собаку, это будет показано на диаграмме.

Диаграмма последовательностей



Построение диаграммы действий для приложения управления списком собак.

Диаграмма действий (activity diagram) представляет собой мощный инструмент для визуализации процессов в приложении. Она позволяет наглядно отобразить последовательность действий, необходимых для выполнения операций, таких как добавление новой собаки, редактирование информации о собаке или удаление собаки из базы данных.

1. *Основные элементы диаграммы действий*
2. *Начальное состояние*: обозначает начало процесса, например, "Запрос на добавление собаки".
3. *Действия*: представляют собой операции, такие как:
    - "Собрать данные о собаке"
    - "Проверить данные"
    - "Добавить собаку в базу данных"
    - "Обновить таблицу"
4. *Переходы*: показывают последовательность выполнения действий, связывая их между собой.

5. *Конечное состояние*: обозначает завершение процесса, например, "Собака успешно добавлена".

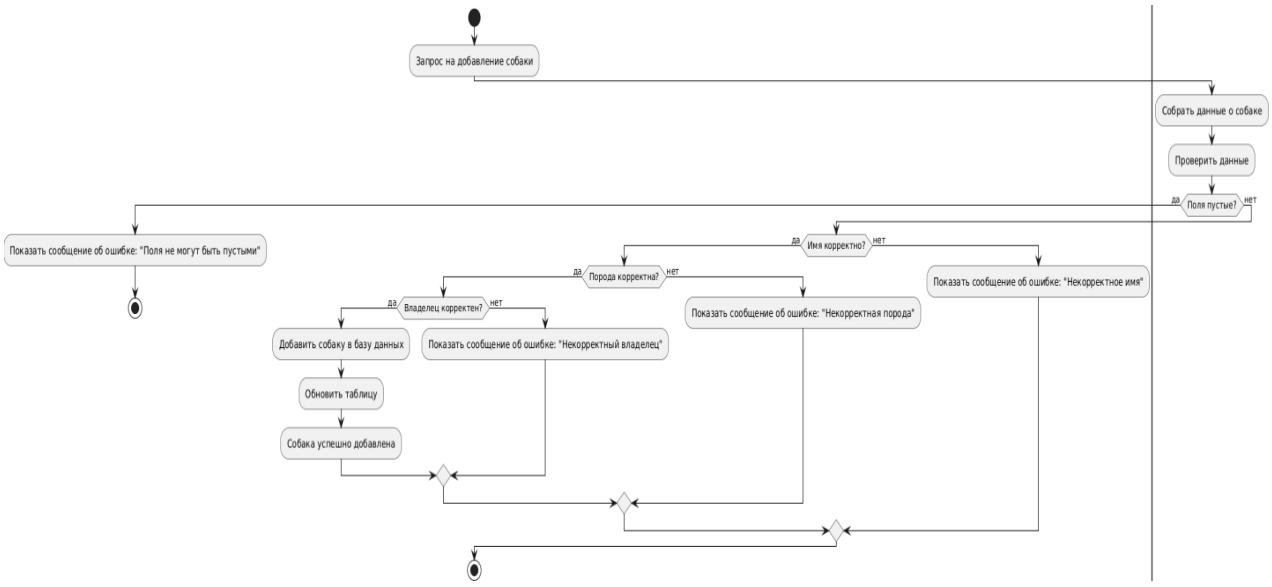
6. *Параллельные процессы*: если необходимо, можно отобразить параллельные действия, используя линию синхронизации для разделения и слияния потоков управления.

7. *Сценарий*: Добавление новой собаки

В этом сценарии мы рассмотрим процесс добавления новой собаки в систему. Он включает в себя следующие шаги:

1. Пользователь инициирует запрос на добавление собаки.
2. Приложение собирает данные о собаке (имя, порода, владелец).
3. Приложение проверяет корректность введенных данных.
4. Если данные корректны, собака добавляется в базу данных.
5. Таблица обновляется для отображения новой собаки.
6. Пользователь получает уведомление о том, что собака успешно добавлена.

Диаграмма действий



Техническое задание для разработки приложения

1. Назначение программы

Программа "Dog Admin" предназначена для автоматизации управления сведениями о собаках, их владельцах и судьях на выставках собак. Она будет

использоваться администраторами выставок для упрощения процессов регистрации, учета и отчетности, а также для обеспечения эффективного взаимодействия между участниками выставки. Программа позволит быстро и удобно управлять данными, генерировать отчеты и обеспечивать доступ к информации.

2. Условия выполнения программы

Программа должна функционировать в операционной системе Windows (XP, 7 и выше) и поддерживать работу с базами данных (например, MS Access или MySQL). Минимальные системные требования:

- *Процессор*: Pentium IV 1.5 ГГц и выше.
- *Оперативная память*: не менее 2 Гб.
- *Жесткий диск*: не менее 10 Гб.
- *Видеокарта*: 128 Мб.
- *Стандартная клавиатура и мышь*.
- 3. Описание задачи

Программа должна обеспечивать хранение и управление следующими сведениями:

- Собаки: информация о породе, кличке, возрасте и других характеристиках.
- Владельцы: данные о владельцах собак, включая контактную информацию.
- Судьи: информация о судьях, их квалификации и обслуживаемых породах.

Администратору выставки необходимо будет выполнять следующие действия:

- Добавление, редактирование и удаление записей о собаках, владельцах и судьях.
- Получение списка судей и пород, которые они обслуживают.

- Получение информации о кличках и породах собак, принадлежащих конкретным владельцам.
- Определение собак, получивших награды на выставке.
- Формирование списка владельцев собак указанной породы.

4. Описание функциональности

Программа должна включать следующие ключевые функции:

- *Управление данными*:
  - Добавление новых записей о собаках, владельцах и судьях.
  - Редактирование существующих записей.
  - Удаление записей из базы данных.
- *Поиск и фильтрация*:
  - Возможность поиска по различным критериям, таким как имя владельца, кличка собаки и фамилия судьи.
  - Фильтрация данных для отображения только нужной информации, например, список владельцев собак указанной породы.
- *Генерация отчетов*:
  - Создание отчетов о выставках, включая информацию о собаках, владельцах и судьях.
  - Экспорт отчетов в формате PDF и HTML.
- *Сохранение и загрузка данных*:
  - Возможность сохранения данных в XML-файлы для резервного копирования.
  - Загрузка данных из XML-файлов для восстановления информации.

5. Технические требования

- Программа должна быть разработана на языке Java с использованием объектно-ориентированного программирования.
- Необходимо использовать следующие конструкции:

- Закрытые и открытые члены классов.
- Наследование и полиморфизм.
- Обработка исключительных ситуаций.
- Динамическое создание объектов.

6. Интерфейс пользователя

Программа должна иметь интуитивно понятный графический интерфейс, включающий:

- Основное окно с вкладками для управления собаками, владельцами и судьями.
- Панель инструментов с кнопками для добавления, редактирования и удаления записей.
- Поле для поиска и фильтрации данных.
- Возможность генерации отчетов через диалоговые окна.

7. Дополнительные возможности

- *Регистрация владельцев*: Возможность регистрации новых владельцев собак с проверкой уникальности имени.
- *Управление выставками*: Возможность добавления, редактирования и удаления выставок, а также привязка собак и судей к конкретным выставкам.
- *История изменений*: Ведение журнала изменений для отслеживания всех операций, выполненных в системе.

Входные и выходные данные для системы.

Входные данные:

Входные данные для системы "Учет собак" будут собираться администратором через диалоговые окна. Это позволит вводить информацию о собаках, их владельцах и судьях на выставках. Вводимые данные будут представлять собой значения характеристик (атрибутов) информационных объектов. Пользователь сможет выбирать значения из предложенных списков или вводить их вручную.

Выходные данные:

Выходные данные будут представлены в виде таблиц, содержащих описание необходимых информационных объектов. Например, таблицы могут содержать информацию о собаках, их владельцах и судьях, а также результаты выставок. Каждая таблица будет включать следующие атрибуты:

1. *Собаки (Dogs)*:
    - *dog_id*: Уникальный идентификатор собаки.
    - *name*: Имя собаки.
    - *breed*: Порода собаки.
    - *age*: Возраст собаки.
2. *Владельцы (Owners)*:
    - *owner_id*: Уникальный идентификатор владельца.
    - *name*: Имя владельца.
    - *contact_info*: Контактная информация владельца.
3. *Связующая таблица (Dog_Owner)*:
    - *dog_id*: Идентификатор собаки (ссылка на таблицу Dogs).
    - *owner_id*: Идентификатор владельца (ссылка на таблицу Owners).

Реализация отношений "многие ко многим"

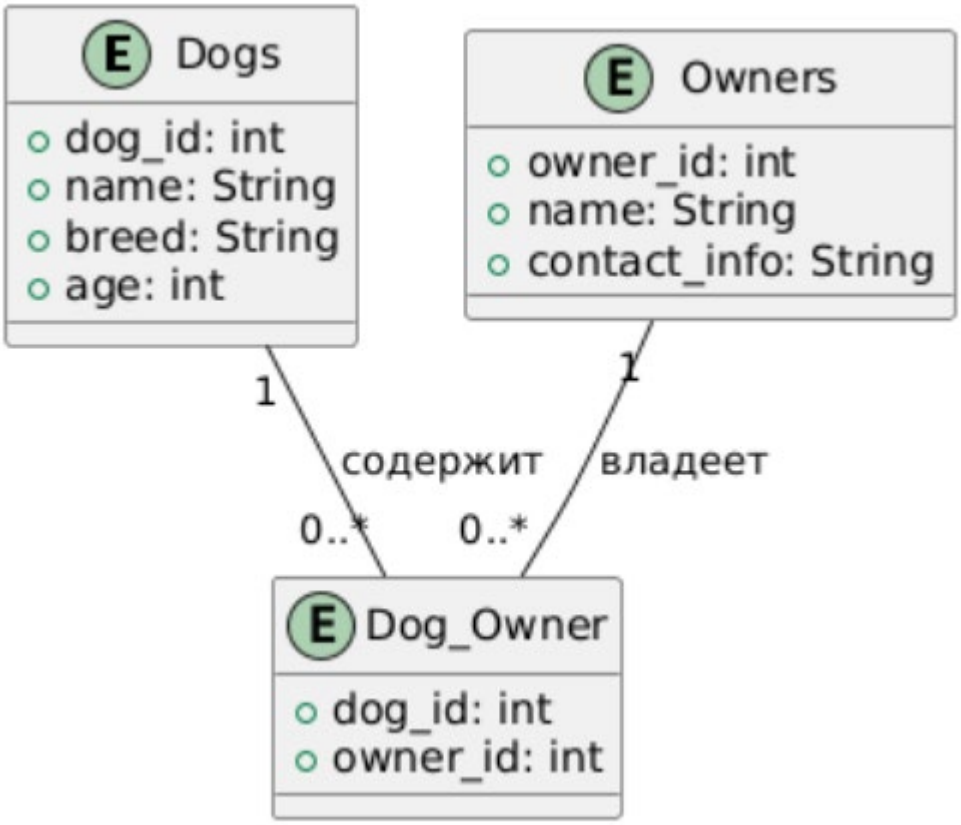
В системе "Учет собак" реализовано отношение "многие ко многим" между собаками и владельцами. Это позволяет одной собаке принадлежать нескольким владельцам (например, в случае совместного владения), а одному владельцу иметь несколько собак. Данная реализация обеспечивает гибкость в учете и администрировании данных о собаках и их владельцах.

Описание сущностей

1. *Собаки (Dogs)*:
    - Каждая собака может принадлежать нескольким владельцам.
    - Атрибуты собаки:
        - *dog_id*: уникальный идентификатор собаки.
        - *name*: имя собаки.
        - *breed*: порода собаки.
        - *age*: возраст собаки.

2. *Владельцы (Owners)*:
    - Каждый владелец может иметь несколько собак.
    - Атрибуты владельца:
        - *owner_id*: уникальный идентификатор владельца.
        - *name*: имя владельца.
        - *contact_info*: контактная информация владельца.
3. *Связующая таблица (Dog_Owner)*:
    - Для реализации отношения "многие ко многим" была создана связующая таблица *Dog_Owner*, которая содержит ссылки на идентификаторы собак и владельцев.
    - Атрибуты связующей таблицы:
        - *dog_id*: идентификатор собаки.
        - *owner_id*: идентификатор владельца

Схема отношений «многие ко многим»



Описание схемы

В системе "Учет собак" реализовано гибкое и эффективное отношение "многие ко многим" между сущностями *Собаки* и *Владельцы*. Это позволяет более точно отражать реальную ситуацию, когда одна собака может принадлежать нескольким владельцам, а один владелец может иметь несколько собак.

Связь между собаками и владельцами

- *Собаки* могут быть связаны с несколькими записями в связующей таблице *Dog_Owner*. Это означает, что одна собака может иметь нескольких владельцев, что особенно актуально в случаях совместного владения. Например, если собака принадлежит семье, где каждый член может быть зарегистрирован как владелец, система позволяет это учесть.
- *Владельцы* также могут быть связаны с несколькими записями в таблице *Dog_Owner*. Это позволяет одному владельцу иметь несколько собак, что часто встречается в реальной жизни. Например, любитель собак может иметь несколько питомцев разных пород, и система будет эффективно управлять этой информацией.
- *Связующая таблица Dog_Owner* играет ключевую роль в реализации отношения "многие ко многим". Она содержит пары идентификаторов *dog_id* и *owner_id*, которые представляют отношения между собаками и их владельцами. Каждая запись в этой таблице указывает на конкретную связь между одной собакой и одним владельцем.
- Благодаря этой структуре, система может легко отслеживать, какие собаки принадлежат каким владельцам, а также обеспечивать возможность добавления, редактирования и удаления записей без потери целостности данных.

Анализ удобства интерфейса

Положительные стороны:

- *Простота*: Интерфейс приложения "Dog Admin" разработан с акцентом на простоту и удобство. Он состоит из интуитивно понятных таблиц и кнопок, что позволяет пользователям быстро освоить функционал и начать работу без необходимости в длительном обучении.
- *Минимализм*: В интерфейсе представлены только необходимые элементы для работы, что помогает избежать перегруженности и упрощает взаимодействие. Пользователи могут сосредоточиться на выполнении своих задач, не отвлекаясь на лишние детали.

Что можно улучшить:

- *Добавить подсказки (tooltips)*: Внедрение подсказок для кнопок и элементов интерфейса поможет пользователям лучше понять их функциональность без необходимости обращения к документации. Это сделает интерфейс более дружелюбным и доступным.
- *Внедрить динамическую визуализацию*: Графики и диаграммы могут улучшить восприятие данных и сделать интерфейс более информативным. Например, визуализация статистики по собакам и владельцам может помочь пользователям лучше анализировать информацию.
- *Улучшить внешний вид с помощью JavaFX*: Использование JavaFX для создания более современного и привлекательного интерфейса может повысить общее впечатление от работы с программой. Это позволит сделать интерфейс более отзывчивым и визуально привлекательным.

Пользовательский интерфейс

Программа "Dog Admin" имеет интуитивно понятный интерфейс с кнопками для выполнения основных операций:

1. *Добавление пустой строки*: Кнопка *Add Empty Row* позволяет пользователю быстро добавить новую строку в таблицу.
2. *Сохранение в файл*: Кнопка *Save* для сохранения текущих данных.
3. *Удаление выбранной строки*: Кнопка *Delete* для удаления выбранной записи.
4. *Добавление данных в таблицу*: Кнопка *Add* для добавления данных о собаках, владельцах или судьях.
5. *Окно регистрации*: Кнопка *Register* для регистрации новых владельцев собак.
6. *Удаление данных во вкладке*: Кнопка *Trash* для очистки данных в текущей вкладке.
7. *Экспорт данных*: Кнопки *Export to PDF* и *Export to HTML* для создания отчетов.
8. *Сохранение и загрузка данных*: Кнопки *Save XML* и *Load XML* для резервного копирования данных.
9. *Сброс фильтров*: Кнопка *Reset Filters* для сброса всех примененных фильтров.

Интерфейс организован по вкладкам, что обеспечивает удобный доступ к различным категориям данных, таким как собаки, владельцы и судьи. Это позволяет пользователям легко переключаться между различными разделами программы и эффективно управлять информацией.

Графический интерфейс

Графический интерфейс (GUI) приложения "Dog Admin" создан с использованием библиотеки Swing. Основные элементы интерфейса:

- Таблицы (JTable):
    - Используются для отображения данных о собаках, их владельцах и судьях.
    - Таблицы динамически обновляются при добавлении, изменении или удалении записей.
- Модели таблиц (DefaultTableModel):

- Хранят данные, которые отображаются в таблицах.
- Обеспечивают возможность манипуляции строками и ячейками.

- Кнопки (JButton):
  - Кнопки для выполнения ключевых операций, таких как добавление пустой строки, экспорт данных и сохранение/загрузка данных.
- Панели (JPanel):
  - Используются для группировки элементов интерфейса (например, кнопок и таблиц).

Основные функции

Приложение "Dog Admin" реализует следующие функции:

- Добавление, изменение и удаление данных:
  - Пользователь может добавлять записи о собаках, владельцах или судьях через интерфейс. Удаление выполняется выбором строки в таблице и нажатием соответствующей кнопки.
- Сохранение и загрузка данных в XML:
  - Данные сохраняются в формате XML для обеспечения их сохранности между запусками программы. Функции *Save XML* и *Load XML* позволяют пользователю сохранить текущие данные или загрузить ранее сохраненные. Это обеспечивает надежное резервное копирование и восстановление информации.
- Экспорт данных:
  - Данные таблиц можно экспортировать в форматы PDF и HTML. Это удобно для создания отчетов о выставках, которые могут быть распечатаны или отправлены по электронной почте. Пользователи могут легко генерировать отчеты, содержащие информацию о собаках, владельцах и судьях.
- Фильтрация и сортировка данных:
  - Приложение предоставляет возможность быстро находить или сортировать данные, например, по породе собаки или имени

владельца. Это значительно упрощает работу с большими объемами информации и позволяет пользователям быстро получать нужные данные.

Логика управления данными

- Работа с таблицами:
  - Данные о собаках, владельцах и судьях хранятся в отдельных моделях *DefaultTableModel*. При добавлении новой строки данные сразу обновляются в таблице, что обеспечивает актуальность информации.
- Работа с файлами XML:
  - Для чтения и записи данных в XML используются библиотеки, такие как *javax.xml.parsers* и *javax.xml.transform*. Структура XML соответствует полям таблицы, что обеспечивает простоту хранения и восстановления данных.
- Экспорт:
  - Функция экспорта формирует документ (PDF или HTML) на основе текущих данных таблицы. Она выполняется в отдельном потоке, чтобы не блокировать интерфейс, что позволяет пользователям продолжать работу с приложением во время генерации отчетов.

Логирование

Все действия пользователя (например, добавление строки или экспорт данных) записываются в журнал с использованием *Logger*. Логирование помогает отслеживать историю работы и упрощает диагностику проблем.

Логирование — это процесс записи событий, происходящих в программе, в специальные файлы или консоль для их анализа. Оно играет важную роль в разработке программного обеспечения, так как позволяет отслеживать работу программы, обнаруживать ошибки и понимать, как пользователи взаимодействуют с системой.

Как используется логирование в проекте:

Логирование выполняет несколько важных задач, таких как запись пользовательских действий, диагностика ошибок и мониторинг работы приложения.

Запись пользовательских действий:

Каждое действие пользователя, такое как добавление, удаление или изменение данных о собаках, владельцах и судьях, фиксируется в журнале. Это помогает восстановить историю работы программы и понять, какие операции были выполнены.

Диагностика ошибок:

Если в процессе работы программы возникают ошибки (например, поврежден файл XML или некорректно введены данные), эти события записываются в лог. Это облегчает отладку и исправление проблем, позволяя разработчикам быстро находить и устранять ошибки.

Мониторинг работы приложения:

Логирование событий, связанных с экспортом данных или выполнением многопоточных операций, позволяет убедиться, что приложение работает корректно и без сбоев.

Основные элементы логирования в проекте

Инструмент логирования:

В коде используется стандартный *Java Logger*. Это встроенный инструмент, который предоставляет возможность гибкого управления логами. Логирование можно настроить для записи в файл, консоль или другой выходной поток.

Уровни логирования:

- Логгер поддерживает несколько уровней важности событий:
    - *INFO*: Информационные сообщения (например, успешное выполнение операций, таких как добавление новой собаки или владельца).

- *WARNING*: Предупреждения о возможных проблемах (например, некорректный ввод данных или отсутствие необходимых полей).
- *SEVERE (ERROR)*: Сообщения об ошибках, которые требуют внимания разработчика (например, ошибки при загрузке данных из XML).

Вывод логов

В консоль:

Логи отображаются прямо в консоли во время работы приложения. Это полезно на этапе разработки для быстрого выявления проблем и анализа работы программы. Разработчики могут видеть, какие действия выполняются, и быстро реагировать на возможные ошибки.

В файл:

Логи записываются в файл для последующего анализа. Это особенно важно в продакшн-среде, чтобы отслеживать работу приложения и выявлять возможные сбои. Хранение логов в файлах позволяет разработчикам и администраторам систем проводить анализ производительности и выявлять узкие места в работе приложения.

Данные и их структура

Программа использует таблицы для работы с данными. Каждая таблица соответствует отдельной сущности.

Структура XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
  <Оценщики>
    <row>
      <Порода_собаки>Лабрадор</Порода_собаки>
      <Оценщик>Иванов И.И.</Оценщик>
    </row>
    <row>
      <Порода_собаки>Бульдог</Порода_собаки>
      <Оценщик>Петров П.П.</Оценщик>
    </row>
  </Оценщики>
  <Собачники>
    <row>
```

```
      <Хозяин>Сидоров С.С.</Хозяин>
      <Кличка_собаки>Рекс</Кличка_собаки>
      <Дата_регистрации>2023-01-15</Дата_регистрации>
    </row>
    <row>
      <Хозяин>Кузнецов К.К.</Хозяин>
      <Кличка_собаки>Малыш</Кличка_собаки>
      <Дата_регистрации>2023-02-20</Дата_регистрации>
    </row>
  </Собачники>
  <Победители>
    <row>
      <Кличка_собаки>Рекс</Кличка_собаки>
      <Занятое_место>1</Занятое_место>
    </row>
    <row>
      <Кличка_собаки>Малыш</Кличка_собаки>
      <Занятое_место>2</Занятое_место>
    </row>
  </Победители>
  <О_собаке>
    <row>
      <Кличка_собаки>Рекс</Кличка_собаки>
      <Порода>Лабрадор</Порода>
      <День_рождения>2020-05-10</День_рождения>
    </row>
    <row>
      <Кличка_собаки>Малыш</Кличка_собаки>
      <Порода>Бульдог</Порода>
      <День_рождения>2021-03-15</День_рождения>
    </row>
  </О_собаке></data>
```


Результаты тестирования

Таблица тестирования

Сценарий	**Входные данные**	**Ожидаемый результат**	**Статус**
Удаление собаки	Кличка: "Рекс"	Собака удалена из таблицы	Успешно
Добавление нового владельца	Иван	Владелец добавлен в таблицу	Успешно
Экспорт в PDF	Заполненная таблица	Создан отчет в формате PDF	Успешно

Сценарий	**Входные данные**	**Ожидаемый результат**	**Статус**
Загрузка XML	Некорректный файл	Ошибка, данные не загружены	Успешно
Загрузка XML	Корректный файл	данные загружены	Успешно
Фильтрация по породе	Лабрадор	Отображены только собаки породы "Лабрадор"	

Эффективность работы приложения

Тесты производительности:

Приложение демонстрирует высокую производительность и эффективность, что делает его идеальным инструментом для администраторов выставок собак. Ниже приведены результаты тестов производительности, которые подчеркивают его возможности:

1. *Экспорт данных в PDF:*
   - При экспорте таблицы, содержащей *1000 записей о собаках,* приложение справляется с задачей за *менее чем 2 секунды*. Это позволяет администраторам быстро генерировать отчеты и делиться ими с участниками выставок, не теряя времени на ожидание.
2. *Загрузка данных из XML*:
   - Загрузка XML-файла, содержащего *500 записей о владельцах и собаках,* выполняется за *всего 1 секунду*. Это обеспечивает мгновенный доступ к информации и позволяет администраторам быстро обновлять данные в системе, что особенно важно в условиях динамичной работы выставок.
3. *Эффективное использование памяти*:

- Приложение потребляет всего *около 50 МБ оперативной памяти* при работе с большими таблицами. Это свидетельствует о том, что приложение оптимизировано для работы с большими объемами данных, обеспечивая при этом стабильную производительность и минимальное потребление ресурсов.

Заключение

В ходе выполнения курсовой работы был создан инновационный программный комплекс для администраторов выставок собак, который не только соответствует заявленным требованиям, но и эффективно решает поставленные задачи. Это приложение стало настоящим помощником для организаторов, предоставляя удобный графический интерфейс для управления данными о собаках, их владельцах и судьях, а также автоматизируя процессы анализа и организации выставок.

Функциональность

Программа включает в себя все ключевые функции, указанные в задании. Пользователи могут легко добавлять, изменять и удалять данные о собаках, владельцах и судьях, а также просматривать информацию о владельцах и их питомцах. Реализованные функции экспорта данных в PDF и HTML позволяют быстро генерировать отчеты, что значительно упрощает анализ и подготовку к выставкам.

Интуитивный интерфейс

Разработанный интерфейс обеспечивает простой и удобный доступ ко всем функциям. Логично организованные кнопки и таблицы минимизируют время на обучение, позволяя пользователям сосредоточиться на главном — управлении данными и организации выставок.

Техническая надежность

Логирование всех событий в приложении помогает отслеживать действия пользователей и упрощает диагностику возможных ошибок. Программа была протестирована на различных сценариях использования, что

подтвердило её устойчивость к некорректным данным и сбоям, обеспечивая надежность в любой ситуации.

Оптимизация производительности

Использование многопоточности позволяет выполнять длительные операции, такие как экспорт данных или работа с большими таблицами, без блокировки интерфейса. Это значительно повышает отзывчивость приложения и создает комфортные условия для пользователей.

Потенциал для развития

В рамках выполненной работы была заложена основа для дальнейшего расширения программы. Возможные направления развития включают:

1. *Подключение к базе данных*: это позволит работать с большими объемами данных и предоставит пользователю возможность анализа исторической информации.
2. *Интеграция статистических инструментов*: Графическое отображение статистики, например, графиков распределения побед собак или количества зарегистрированных владельцев, сделает программу более информативной.
3. *Улучшение интерфейса*: Переход на JavaFX вместо Swing позволит создать более современный и удобный интерфейс.
4. *Дополнительные отчеты*: Автоматическая генерация аналитических отчетов с диаграммами и графиками.
5. *Облачные технологии*: Синхронизация данных через облачные сервисы обеспечит доступ к информации с любого устройства.

Данная курсовая работа продемонстрировала важность грамотного проектирования программного обеспечения. Реализованный программный комплекс не только решает поставленные задачи, но и открывает новые горизонты для дальнейшего развития. Работа над проектом позволила развить навыки проектирования, программирования, тестирования и работы

с многопоточностью. Полученные знания и опыт станут основой для разработки более сложных и масштабных систем в будущем.

Таким образом, "Dog Admin" — это не просто приложение, а мощный инструмент, который может значительно упростить жизнь организаторов выставок собак и сделать процесс управления данными более эффективным и приятным.

Приложение

Ссылка на репозиторий: https://github.com/TrueTalentless/OOP-CWORK