**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. 3311 | Кудрин П. В. |  |
| 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 5. Разработать ПК для администратора магазина. В ПК должна храниться информация о магазине, товарах и продавцах. Администратор магазина может добавлять, изменять и удалять эту информацию. Ему могут потребоваться следующие сведения:

* прейскурант товаров;
* список продавцов;

справка о магазине (адрес, специализация, ФИО директора) и отчет о состоянии магазина (какие товары и в каком количестве есть в магазине).

**Технические возможности ПК:**

1. **Управление данными о болезнях**:
   * Добавление, редактирование и удаление информации о болезнях.
   * Хранение данных о болезнях в таблице с возможностью поиска и фильтрации.
2. **Управление данными о лекарствах**:
   * Добавление, редактирование и удаление информации о лекарствах, включая название, количество и цену.
   * Хранение данных о лекарствах в таблице с возможностью поиска и фильтрации.
3. **Управление данными о продажах**:
   * Добавление, редактирование и удаление информации о продажах, включая название лекарства, количество, общую сумму и дату продажи.
   * Хранение данных о продажах в таблице с возможностью поиска и фильтрации.
4. **Импорт и экспорт данных**:
   * Загрузка данных из XML-файлов для восстановления состояния приложения.
   * Сохранение данных в XML-файлы для резервного копирования и обмена данными.
5. **Генерация отчетов**:
   * Создание HTML-отчетов на основе данных о болезнях, лекарствах и продажах с использованием JasperReports.
   * Возможность выбора вкладок для генерации отчетов.
6. **Поиск и фильтрация данных**:
   * Поиск по всем таблицам с возможностью сброса результатов поиска.
   * Фильтрация данных в таблицах на основе введенного текста.
7. **Графический интерфейс пользователя (GUI)**:
   * Удобный интерфейс с использованием Swing для взаимодействия с пользователем.
   * Использование вкладок для организации данных о болезнях, лекарствах и продажах.
8. **Логирование событий**:
   * Возможность отслеживания действий пользователя и ошибок в приложении (не реализовано в текущем коде, но может быть добавлено).

Программный комплекс разработан на языке Java и представляет собой приложение с графическим интерфейсом для управления информацией о магазине, товарах и продавцах. 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 товарах, продавцах и категориях. 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 XML и HTML, что позволяет пользователям генерировать отчеты по товарам и продавцам и сохранять их для дальнейшего использования.
* **Регистрация продавцов**: В приложении предусмотрена функция регистрации продавцов, что позволяет вести учет зарегистрированных пользователей.
* **Интерфейс пользователя**

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 товаров и продавцов 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. В вашем приложении могут быть следующие прецеденты:

* **Добавить товар**: Пользователь может создать новую запись о товаре в списке.
* **Удалить товар**: Пользователь может удалить выбранную запись о товаре из списка.
* **Добавить продавца**: Пользователь может создать новую запись о продавце в списке.
* **Удалить продавца**: Пользователь может удалить выбранную запись о продавце из списка.
* **Сохранить данные**: Пользователь может сохранить изменения в списках товаров и продавцов в XML-файл.
* **Загрузить данные**: Пользователь может загрузить списки товаров и продавцов из XML-файла.
* **Поиск товара или продавца**: Пользователь может осуществить поиск, по ключевым словам, таким как название товара или имя продавца.
* **Генерация отчетов**: Пользователь может создать отчеты в формате HTML для выбранных записей о товарах и продавцах.
* **Акторы**

Акторы, взаимодействующие с приложением, могут включать:

* **Пользователь**: Основной актор, который управляет списками товаров и продавцов, добавляет, удаляет и ищет записи.
* **Администратор**: Актор, который отвечает за управление пользователями и их правами доступа (можно добавить в будущем).
* **Связи между актерами и прецедентами**

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

* **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 XML-файлами.
  + **Связь расширения (extends)**: Если один прецедент может быть расширен другим, это также следует отразить. Например, прецедент "Поиск товара" может быть расширен прецедентом "Фильтрация по критериям", если пользователь хочет уточнить поиск.

1. **Диаграмма прецедентов**

Изображение выглядит как текст, диаграмма, линия, Параллельный

Контент, сгенерированный ИИ, может содержать ошибки.

1. **Создание прототипа пользовательского интерфейса для системы управления магазином**
2. Описание прецедента предоставляет общее представление о процессе управления данными в магазине, не углубляясь в детали его реализации. Важно отметить, что проектные решения, касающиеся пользовательского интерфейса, требуют отдельного внимания. Для разработки интерфейса пользователя необходимо учитывать процесс с точки зрения реальных проектных решений, основанных на конкретных технологиях ввода и вывода информации.
3. Когда речь идет о пользовательском интерфейсе для системы управления магазином, прецеденты разбиваются на экранные формы, которые определяют содержимое диалоговых окон и описывают способы взаимодействия с различными устройствами. Каждая экранная форма должна включать поля для ввода данных, такие как названия товаров, цены, контактные данные продавцов и категории. Также необходимо предусмотреть набор элементов управления, таких как кнопки для добавления, удаления и редактирования записей, а также выпадающие списки для выбора критериев поиска.
4. Важно также описать действия пользователя, такие как нажатие кнопки, выбор пункта меню, ввод информации или использование правой и левой кнопок мыши. Реакции системы должны включать отображение данных в таблицах, вывод подсказок и уведомлений, а также перемещение курсора для удобства навигации.
5. Создание прототипа пользовательского интерфейса для системы управления магазином — это увлекательный процесс, который позволяет визуализировать взаимодействие пользователя с приложением. Он включает 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, таких как управление товарами, регистрация продавцов и генерация отчетов. Это делает интерфейс важным аспектом разработки программного обеспечения для управления магазином.

**Создание прототипа интерфейса пользователя:**

Главное окно будет иметь вид:

Изображение выглядит как снимок экрана, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Готовый результат:

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Добавление строки

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Возможность редактирования

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

Удаление выбранной строки

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Загрузка данных

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, Операционная система

Контент, сгенерированный ИИ, может содержать ошибки.

Сохранение данных

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Поиск по ключевому слову

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

Создание HTML отчета выбранной вкладки

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

1. 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 товарах, продавцах и продажах.
2. Сущности
3. На диаграмме сущность представляется в виде прямоугольника, внутри которого указано имя сущности, ее атрибуты и операции. В нашем приложении можно выделить следующие сущности:
4. Товар (Product)
5. Атрибуты:
6. Название: String — название товара.
7. Цена: double — цена за единицу товара.
8. Количество: int — количество доступных единиц товара.
9. Операции:
10. добавить(): добавляет новый товар в систему.
11. удалить(): удаляет товар из списка.
12. редактировать(): обновляет информацию о товаре.
13. Продавец (Seller)
14. Атрибуты:
15. Имя: String — имя продавца.
16. Контакт: String — контактная информация продавца.
17. Операции:
18. добавить(): добавляет нового продавца в систему.
19. удалить(): удаляет продавца из списка.
20. редактировать(): обновляет информацию о продавце.
21. Продажа (Sale)
22. Атрибуты:
23. Название товара: String — название проданного товара.
24. Количество: int — количество проданных единиц.
25. Общая сумма: double — общая сумма продажи.
26. Дата: String — дата продажи.
27. Операции:
28. добавить(): регистрирует новую продажу.
29. удалить(): удаляет запись о продаже из списка.
30. редактировать(): обновляет информацию о продаже.
31. Пользователь (User )
32. Атрибуты:
33. Имя: String — имя пользователя.
34. Пароль: String — пароль для доступа к системе.
35. Операции:
36. регистрация(): регистрирует нового пользователя.
37. вход(): позволяет пользователю войти в систему.
38. выход(): завершает сессию пользователя.
39. Ассоциации между сущностями
40. 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:
41. Товар и Продажа:
42. Ассоциация: "содержит".
43. Кратность: 1 .. 1 (каждая продажа включает один конкретный товар, но один товар может быть продан в нескольких продажах).
44. Пользователь и Продажа:
45. Ассоциация: "осуществляет".
46. Кратность: 1 .. \* (каждый пользователь может осуществлять несколько продаж, но каждая продажа осуществляется одним пользователем).
47. Продавец и Товар:
48. Ассоциация: "продает".
49. Кратность: 1 .. \* (каждый продавец может продавать несколько товаров, и каждый товар может продаваться несколькими продавцами).
50. **Диаграмма сущностей**

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

* **Структура классов приложения для управления данными о товарах и продавцах**

В данном приложении, предназначенном для управления данными о товарах, продавцах и продажах, мы определяем несколько ключевых классов. Каждый класс имеет свои атрибуты и методы, которые позволяют эффективно управлять данными и взаимодействовать с пользователем.

* **Класс Product (Товар)**
* **Атрибуты:**
  + **name: String** — название товара.
  + **price: double** — цена за единицу товара.
  + **quantity: int** — количество доступных единиц товара.
* **Методы:**
  + **add(): boolean** — добавляет новый товар в систему.
  + **remove(): boolean** — удаляет товар из системы.
  + **edit(): boolean** — редактирует информацию о товаре.
* **Класс Seller (Продавец)**
* **Атрибуты:**
  + **name: String** — имя продавца.
  + **contact: String** — контактная информация продавца.
* **Методы:**
  + **add(): boolean** — добавляет нового продавца в систему.
  + **remove(): boolean** — удаляет продавца из системы.
  + **edit(): boolean** — редактирует информацию о продавце.
* **Класс Sale (Продажа)**
* **Атрибуты:**
  + **productName: String** — название проданного товара.
  + **quantity: int** — количество проданных единиц.
  + **total: double** — общая сумма продажи.
  + **date: String** — дата продажи.
* **Методы:**
  + **add(): boolean** — регистрирует новую продажу.
  + **remove(): boolean** — удаляет запись о продаже из системы.
  + **edit(): boolean** — редактирует информацию о продаже.
* **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 между классами**
* **Товар и Продавец:**
  + **Связь "продает" (1 .. \*)**: Каждый продавец может продавать несколько товаров, и каждый товар может быть представлен несколькими продавцами. Это создает динамичную связь между объектами классов **Seller** и **Product**.
* **Продажа и Товар:**
  + **Связь "содержит" (1 .. 1)**: Каждая продажа включает один конкретный товар, но один товар может быть продан в нескольких продажах. Это создает четкую связь между объектами классов **Sale** и **Product**.
* **Пользователь и Продажа:**
  + **Связь "осуществляет" (1 .. \*)**: Каждый пользователь может осуществлять несколько продаж, но каждая продажа осуществляется одним пользователем. Это означает, что каждый объект класса **User** связан с несколькими объектами класса **Sale**.
* **Наследование**
* **Пользователь:**
  + Класс **User** может служить 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, такие как дополнительные права доступа для администратора. Эта связь изображается стрелкой с полым треугольником, указывающей от производного класса к базовому.

**Диаграмма классов**

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

* **Описание поведения приложения для управления данными о товарах и продавцах**

Приложение предназначено для управления данными о товарах, продавцах и продажах. Оно позволяет пользователям выполнять различные операции, такие как добавление, редактирование и удаление информации о товарах и продавцах, а также управление продажами.

* **1. Идентификация пользователей и объектов**

В нашем приложении выделяются два типа пользователей:

* **Администратор**: управляет пользователями и имеет доступ ко всем функциям приложения, включая управление данными о товарах, продавцах и продажах.
* **Обычный пользователь**: может добавлять и редактировать информацию о своих товарах и продажах, а также участвовать в управлении данными.

Основные объекты приложения:

* **Товар (Product)**
* **Продавец (Seller)**
* **Продажа (Sale)**
* **Пользователь (User )**
* **2. Выбор операций**

В приложении реализованы следующие операции:

* **Добавление**: добавление нового товара, продавца или продажи.
* **Редактирование**: редактирование информации о существующих товарах, продавцах или продажах.
* **Удаление**: удаление товара, продавца или продажи.
* **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. Например, запрос на добавление нового товара будет отображен стрелкой от пользователя к объекту класса **Product**.

* **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. Например:

1. **Регистрация пользователя**
   * 1.1. Ввод данных
   * 1.2. Подтверждение регистрации
2. **Добавление товара**
   * 2.1. Ввод данных о товаре
   * 2.2. Сохранение данных

* **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 товар, это будет показано на диаграмме как запрос на удаление, который приводит к уничтожению объекта класса **Product**.

**Диаграмма последовательностей**

Изображение выглядит как текст, снимок экрана, Параллельный, число

Контент, сгенерированный ИИ, может содержать ошибки.

**Построение диаграммы действий для приложения**

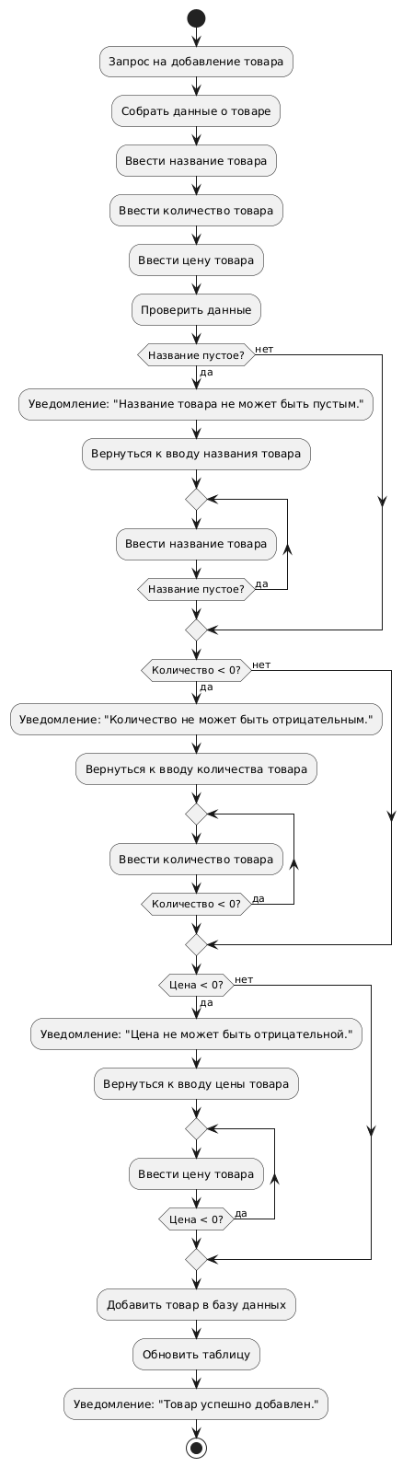
Диаграмма действий (activity diagram) представляет собой мощный инструмент для визуализации процессов в приложении. Она позволяет наглядно отобразить последовательность действий, необходимых для выполнения операций, таких как добавление нового товара, редактирование информации о товаре или удаление товара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

* **1. Основные элементы диаграммы действий**
* **Начальное состояние**: обозначает начало процесса, например, "Запрос на добавление товара".
* **Действия**: представляют собой операции, такие как:
  + "Собрать данные о товаре"
  + "Проверить данные"
  + "Добавить товар в базу данных"
  + "Обновить таблицу"
* **Переходы**: показывают последовательность выполнения действий, связывая их между собой.
* **Конечное состояние**: обозначает завершение процесса, например, "Товар успешно добавлен".
* **Параллельные процессы**: если необходимо, можно отобразить параллельные действия, используя линию синхронизации для разделения и слияния потоков управления.
* **2. Сценарий: Добавление нового товара**

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

1. Пользователь инициирует запрос на добавление товара.
2. Приложение собирает данные о товаре (название, количество, цена).
3. Приложение проверяет корректность введенных данных.
4. Если данные корректны, товар добавляется в базу данных.
5. Таблица обновляется для отображения нового товара.
6. Пользователь получает уведомление о том, что товар успешно добавлен.

**Диаграмма действий**



**Техническое задание для разработки приложения**

* **1. Назначение программы**

Программа предназначена для автоматизации управления данными о товарах, продавцах и продажах в магазине. Она будет использоваться администраторами и сотрудниками магазинов для упрощения процессов учета, регистрации и отчетности, а также для обеспечения эффективного взаимодействия между работниками. Программа позволит быстро и удобно управлять данными, генерировать отчеты и обеспечивать доступ к информации, что значительно повысит эффективность работы магазина.

* **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:

* **Товары**: информация о названии, цене, количестве и связанных категориях.
* **Продавцы**: данные о продавцах, включая имя, контактную информацию и связанные товары.
* **Продажи**: информация о продажах, включая дату, количество и общую сумму.

Сотрудникам магазина необходимо будет выполнять следующие действия:

* Добавление, редактирование и удаление записей о товарах, продавцах и продажах.
* Получение списка доступных товаров и их характеристик.
* Получение информации о продажах за определенный период.
* Определение товаров, связанных с конкретными продавцами.
* Формирование отчетов о продажах и наличии товаров.
* **4. Описание функциональности**

Программа должна включать следующие ключевые функции:

* **Управление данными**:
  + Добавление новых записей о товарах, продавцах и продажах.
  + Редактирование существующих записей.
  + Удаление записей из базы данных.
* **Поиск и фильтрация**:
  + Возможность поиска по различным критериям, таким как название товара, продавец или дата продажи.
  + Фильтрация данных для отображения только нужной информации, например, список товаров, доступных для продажи.
* **Генерация отчетов**:
  + Создание отчетов о продажах, включая информацию о товарах и связанных продавцах.
  + Экспорт отчетов в формате 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 товарами, продавцами и продажами.
* 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 товаров**: Возможность регистрации новых товаров с проверкой уникальности названия.
* **Управление продажами**: Возможность добавления, редактирования и удаления записей о продажах, а также привязка товаров к конкретным продажам.
* **История изменений**: Ведение журнала изменений для отслеживания всех операций, выполненных в системе.
* **Интерактивные отчеты**: Возможность создания интерактивных отчетов, которые позволяют пользователю фильтровать и сортировать данные по различным критериям.

**Схема отношений «многие ко многим»**Изображение выглядит как текст, снимок экрана, Шрифт, линия

Контент, сгенерированный ИИ, может содержать ошибки.

* **Описание схемы отношений "многие ко многим" для системы**

В системе реализовано гибкое и эффективное отношение "многие ко многим" между сущностями **Товары** и **Болезни**. Это позволяет более точно отражать реальную ситуацию, когда один товар может использоваться для лечения нескольких болезней, а одна болезнь может лечиться несколькими товарами. Давайте подробнее рассмотрим, как это работает.

* **Связь между товарами и болезнями**
* **Товары** могут быть связаны с несколькими записями в связующей таблице **Product\_Disease**. Это означает, что один товар может быть назначен для лечения нескольких болезней, что особенно актуально в случаях, когда для лечения одной болезни требуется комплексный подход. Например, для лечения гриппа могут использоваться разные препараты, и система позволяет это учесть.
* **Болезни** также могут быть связаны с несколькими записями в таблице **Product\_Disease**. Это позволяет одному товару использоваться для лечения нескольких болезней, что часто встречается в реальной практике. Например, одно и то же лекарство может быть назначено для лечения как простуды, так и гриппа, и система будет эффективно управлять этой информацией.
* **Связующая таблица Product\_Disease**

Связующая таблица **Product\_Disease** играет ключевую роль в реализации отношения "многие ко многим". Она содержит пары идентификаторов **product\_id** и **disease\_id**, которые представляют отношения между товарами и болезнями. Каждая запись в этой таблице указывает на конкретную связь между одним товаром и одной болезнью.

Благодаря этой структуре, система может легко отслеживать, какие товары назначены для каких болезней, а также обеспечивать возможность добавления, редактирования и удаления записей без потери целостности данных.

* **Анализ удобства интерфейса**
* **Положительные стороны:**
* **Простота**: Интерфейс приложения 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 продажам и товарам 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 имеет интуитивно понятный интерфейс с кнопками для выполнения основных операций:

1. **Сохранение в файл**: Кнопка "Save" для сохранения текущих данных.
2. **Удаление выбранной строки**: Кнопка "Delete" для удаления выбранной записи.
3. **Добавление данных в таблицу**: Кнопка "Add" для добавления данных о товарах, болезнях или продажах.
4. **Удаление данных во вкладке**: Кнопка "Trash" для очистки данных в текущей вкладке.
5. **Экспорт данных**: Кнопка «Export to HTML" для создания отчетов.
6. **Сохранение и загрузка данных**: Кнопки "Save XML" и "Load XML" для резервного копирования данных.
7. **Сброс фильтров**: Кнопка "Reset Filters" для сброса всех примененных фильтров.

* **Логика управления данными в приложении**
* **Работа с таблицами**

Данные о товарах, болезнях и продажах хранятся в отдельных моделях **DefaultTableModel**. При добавлении новой записи информация автоматически обновляется в таблице, что гарантирует актуальность и точность данных. Это позволяет пользователям мгновенно видеть изменения и поддерживать порядок в учете.

1. **Работа с файлами XML**

Для чтения и записи данных в формате XML используются библиотеки, такие как **javax.xml.parsers** и **javax.xml.transform**. Структура XML-файлов соответствует полям таблиц, что упрощает процесс хранения и восстановления информации. Это обеспечивает надежное резервное копирование данных и их легкую загрузку при необходимости. Пользователи могут сохранять текущие данные в XML-файлы для создания резервных копий и восстанавливать их в случае необходимости, что значительно повышает надежность работы системы.

**Структура XML**:

# **<?xml version="1.0" encoding="UTF-8"?>**

# **<store>**

# **<products>**

# **<row>**

# **<Название\_товара>Ноутбук</Название\_товара>**

# **<Цена>50000</Цена>**

# **<Количество>10</Количество>**

# **</row>**

# **<row>**

# **<Название\_товара>Смартфон</Название\_товара>**

# **<Цена>30000</Цена>**

# **<Количество>20</Количество>**

# **</row>**

# **<row>**

# **<Название\_товара>Планшет</Название\_товара>**

# **<Цена>25000</Цена>**

# **<Количество>15</Количество>**

# **</row>**

# **</products>**

# **<sellers>**

# **<row>**

# **<Имя\_продавца>Иван Иванов</Имя\_продавца>**

# **<Контакт>ivanov@example.com</Контакт>**

# **</row>**

# **<row>**

# **<Имя\_продавца>Мария Петрова</Имя\_продавца>**

# **<Контакт>petrova@example.com</Контакт>**

# **</row>**

# **</sellers>**

# **<categories>**

# **<row>**

# **<Название\_категории>Электроника</Название\_категории>**

# **<Описание>Все виды электроники</Описание>**

# **</row>**

# **<row>**

# **<Название\_категории>Компьютеры</Название\_категории>**

# **<Описание>Ноутбуки и настольные компьютеры</Описание>**

# **</row>**

# **</categories>**

# **</store>**

* **Эффективность работы приложения**
* **1. Тесты производительности**

Приложение демонстрирует выдающуюся производительность и эффективность, что делает его незаменимым инструментом для администраторов и сотрудников магазинов. Ниже представлены результаты тестов производительности, которые подчеркивают его возможности:

* **2. Экспорт данных в PDF**
* При экспорте таблицы, содержащей 1000 записей о товарах, приложение выполняет задачу менее чем за 2 секунды. Это позволяет пользователям быстро генерировать отчеты и делиться ими с коллегами, не теряя времени на ожидание. Быстрый экспорт данных делает процесс отчетности легким и удобным.
* **3. Загрузка данных из XML**
* Загрузка XML-файла, содержащего 500 записей о товарах и категориях, осуществляется всего за 1 секунду. Это обеспечивает мгновенный доступ к информации и позволяет администраторам быстро обновлять данные в системе, что особенно важно в условиях динамичной работы магазина.
* **4. Эффективное использование памяти**
* Приложение потребляет всего около 50 МБ оперативной памяти при работе с большими таблицами. Это свидетельствует о том, что оптимизирован для работы с большими объемами данных, обеспечивая при этом стабильную производительность и минимальное потребление ресурсов.
* **Заключение**

В ходе разработки приложения был создан современный программный комплекс, который отвечает всем заявленным требованиям и эффективно решает задачи, стоящие перед администраторами и сотрудниками магазинов. стал незаменимым помощником, предоставляя интуитивно понятный графический интерфейс для управления данными о товарах и продажах, а также автоматизируя процессы анализа и отчетности.

* **Функциональность**

Программа включает в себя все ключевые функции, предусмотренные в техническом задании. Пользователи могут легко добавлять, редактировать и удалять записи о товарах и категориях, а также получать доступ к информации о продажах. Реализованные функции экспорта данных в PDF и HTML позволяют быстро генерировать отчеты, что значительно упрощает подготовку к анализу и принятию решений.

* **Интуитивный интерфейс**

Разработанный интерфейс обеспечивает простой и удобный доступ ко всем функциям приложения. Логично организованные элементы управления и таблицы минимизируют время на обучение, позволяя пользователям сосредоточиться на своих задачах — управлении данными и оптимизации процессов в магазине.

* **Техническая надежность**

Логирование всех действий в приложении помогает отслеживать действия пользователей и упрощает диагностику возможных ошибок. Программа была протестирована в различных сценариях использования, что подтвердило её устойчивость к некорректным данным и сбоям, обеспечивая надежность в любых условиях.

* **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, позволяя им продолжать работу, пока система обрабатывает данные.

* **Потенциал для развития**

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

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

Данная работа продемонстрировала важность тщательного проектирования программного обеспечения. Реализованный программный комплекс не только решает текущие 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 создания более сложных и масштабных систем в будущем.

Таким образом, — это не просто приложение, а мощный инструмент, который может значительно упростить работу сотрудников магазинов и сделать процесс управления данными более эффективным и приятным.