

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(институт)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(кафедра)

**КУРСОВАЯ РАБОТА (КУРСОВОЙ ПРОЕКТ)**

по учебному курсу «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

Вариант \_\_\_\_ *(при наличии)*

|  |  |  |
| --- | --- | --- |
| Студент | (И.О. Фамилия) |  |
| Группа | (И.О. Фамилия) |  |
| Преподаватель | (И.О. Фамилия) |  |

Тольятти 20\_\_

Лист задания

# Содержание

[ВВЕДЕНИЕ 4](#_Toc53570998)

[1 ОПИСАНИЕ ЗАДАЧИ НА ИССЛЕДОВАНИЕ И ЕЕ ПРОЕКТИРОВАНИЕ 6](#_Toc53570999)

[1.1 Формулировка и описание задачи курсового проекта 6](#_Toc53571000)

[1.2 Разработка требования к программному продукту 7](#_Toc53571001)

[1.3 Проектирование программного продукта 9](#_Toc53571002)

[2 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОДУКТА 12](#_Toc53571003)

[2.1 Выбор инструментов разработки 12](#_Toc53571004)

[2.2 Разработка базы данных 13](#_Toc53571005)

[2.3 Описание классов программы 16](#_Toc53571006)

[3 ТЕСТИРОВАНИЕ И РАЗВЁРТЫВАНИЕ ПРИЛОЖЕНИЯ 22](#_Toc53571007)

[3.1 Разработка плана тестирования 22](#_Toc53571008)

[3.2 Тестирование приложение 23](#_Toc53571009)

[3.3 Развертывание приложения 24](#_Toc53571010)

[ЗАКЛЮЧЕНИЕ 26](#_Toc53571011)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc53571012)

[ПРИЛОЖЕНИЕ А. КОД ПРОГРАММЫ 28](#_Toc53571013)

[ПРИЛОЖЕНИЕ Б. КОД СОЗДАНИЯ БАЗЫ ДАННЫХ 104](#_Toc53571014)

[ПРИЛОЖЕНИЕ В. ДИАГРАММА КЛАССОВ. ЧАСТЬ 1 106](#_Toc53571015)

[ПРИЛОЖЕНИЕ Г. ДИАГРАММА КЛАССОВ. ЧАСТЬ 2 107](#_Toc53571016)

# ВВЕДЕНИЕ

Склады являются важной частью технического процесса предприятия, они являются основой оптовой и розничной торговли. Именно поэтому современным организациям, использующим новейшие технологии и располагающим квалифицированными кадрами, необходимо эффективно организовывать складские помещения предприятия, стараясь оставаться впереди конкурентов.

Тема курсового проекта: "система управления оптовым складом".

Важность разработки автоматизированного программного обеспечения для учета складских грузов заключается в том, что сотрудники теперь проводят большую часть своего времени, работая над различными документами и отчетами. Именно поэтому деятельность людей становится все более актуальной благодаря развитию специализированных компьютерных технологий. Разработка программного обеспечения для автоматизированного складского учета грузов позволит повысить эффективность работы сотрудников склада, оперативно вводить необходимую информацию в компьютер и давать подробные ответы о любом поступившем товаре. Это также позволяет сократить вашу работу с организацией документов сотрудников, храня большие объемы информации в базе данных, и вы можете быстро найти любую интересующую вас информацию за считанные секунды.

Работа в других отделах (продажи, доставка, снабжение и т.д.).), Уровень логистических услуг, предоставляемых заказчику, следовательно, уровень конкурентоспособности компании на рынке зависит от того, насколько эффективно будет осуществляться операция на складе. В связи с этим перед складским управлением каждого предприятия стоит задача оптимизации максимального количества технологических элементов.

Объектом исследования является автоматизированный процесс скда.

Предметом исследования явилась автоматизированная система управления оптовым складом.

Целью данного курсового проекта является изучение склада для приема хранящихся товаров и их дальнейшей реализации, а также разработка программы автоматизации склада.

Проблемы автоматизации бухгалтерского учета и складского учета являются одними из наиболее экономических проблем в оперативных исследованиях, и их решение имеет очень важное значение для автоматизации. Определение оптимальной стратегии управления складом правильно и своевременно позволяет высвободить большие объемы денежных ресурсов в виде товаров, тем самым повышая эффективность используемых ресурсов.

В различных складских операциях, с точки зрения складского учета, учет грузов является наиболее трудоемким. Одной из основных задач складского учета является правильная организация данных таким образом,чтобы можно было получить информацию о ходе поступления товаров, поставщиках и получателях продукции для выполнения своих обязательств.

Основным направлением при рассмотрении вопроса складских операций будет являться поступление товаров, а также их реализация.

При решении задач снабжения работники склада должны изучить и учесть спрос и предложение всех ресурсов, потребляемых предприятием, для них и посреднических организаций уровень обслуживания и изменения цен, выбрать наиболее экономичную форму движения, оптимизировать товарно-материальные запасы, снизить затраты на приобретение товаров и складов.

Создание автоматизированной системы позволит систематизировать процесс управления производственными процессами, повысить эффективность взаимодействия между подразделениями, снизить количество ошибок, сократить время, затрачиваемое на основные операции, сократить количество бумажных документов, а также позволит централизованно хранить необходимую информацию.

# ОПИСАНИЕ ЗАДАЧИ НА ИССЛЕДОВАНИЕ И ЕЕ ПРОЕКТИРОВАНИЕ

## Формулировка и описание задачи курсового проекта

Цель курсовой работы - для разработка программного обеспечения для автоматизации учета товаров на оптовых складах.

Необходимо разработать автоматизированную систему, которая будет принимать все складские бухгалтерские операции.

Склад должен выполнять следующие функции: принимать, обрабатывать, хранить и отгружать готовую продукцию, принимать готовую продукцию, организовывать, подбирать потребителей, определять потребности в транспортных средствах, механизированной погрузочной технике, отгрузке продукции в контейнеры и рабочей силе, координировать планирование и подготовку производственных отчетов, а также участвовать в рассмотрении претензий, поступающих на предприятие [1].

Затем склад готовой продукции должен обеспечить создание условий для сохранения товаров,помещенных на временное хранение, организовать эффективное хранение, внутреннюю транспортировку, упаковку и подготовку отгрузки продукции, обеспечить сохранность продукции, подготовить справки о состоянии запасов готовой продукции, подготовить документы (записи, справки, справки, письма) о повреждении продукции, обеспечить высокий уровень механизации и автоматизации транспортно-складских операций, использовать компьютерные системы и нормативную базу организаций и организаций охраны труда., Разработка цен хранения продукции со сторонними организациями, закономерностей и методов контроля за хранением готовой продукции.

Склад обязан вести учет временно хранящейся продукции,оформлять карточки, складские помещения, инвентарные запасы и остатки продукции на складе.

Порядок приема продукции на склад:

Когда товар прибывает на склад, сопровождаемый экспедитором и квитанцией;

* Администратор склада проверяет приходную накладную и регистрирует ее в книге приходных документов (накладных) );
* Контродлер проверяют поступающий товар,если все в порядке,забирают его на склад,через экспедирование груза оформляют выписку(документы), товар принимают на хранение;
* Носильщик отвозит товар в пустое складское помещение,а контролер фиксирует место хранения вновь поступившего товара в бухгалтерской книге.

В процессе инвентаризации требуется проведение инвентаризации,которая состоит из таких этапов: сотрудник, ответственный за инвентаризацию товара, сопровождает книгу повторной отправки на склад, ручной осмотр и переписывание данных о товаре и его номере;затем проверяется инвентаризация товара и его номер.

Отгрузка товара со склада проходит следующие этапы:

* Получатель товара представляет счет-фактуру на отгрузку товара;
* Контролер проверяет этот счет и регистрирует его в книге документов покупки;
* Далее контролер дает указание сотруднику склада найти необходимые продукты и отгрузить их;
* Затем получатель продукта проводит проверку продукта, чтобы определить, был ли продукт отгружен и в необходимом количестве;
* Контроллер регистрирует отгрузку товара в бухгалтерском учете;
* Далее контролер выдает сопроводительные документы на отгрузку товара грузополучателю;
* А затем товар отгружается непосредственно техническими средствами.

Формирование отчета о движении документов продукции на складе: сотрудники в этом поле собирают все документы, входящие и исходящие, все учетные данные по операции и товарам;вручную обрабатывают их для формирования файла итогового отчета;вручную обрабатывают их для формирования файла итогового отчета.

Как видно из этой ситуации, ничто на складе не автоматизировано, сотрудник выполняет операции по всем учетным документам, а также учет результатов ручной переоценки. Формирование файла отчета о движ

## Разработка требования к программному продукту

Целью курсового проекта является разработка ПО для автоматизации учета товара на оптовом складе.

Необходимо разработать такую систему, которая будет иметь операции складского учетаю

Требования к разрабатываемому ПО:

1. Общие требования:

* диалоговый режим взаимодействия пользователей с ПО;
* простой и понятный интерфейс.

1. Функциональные требования:

* Просмотр заказов;
* Добавление заказов;
* Удаление заказов;
* Редактирование заказов
* Просмотр поставок;
* Добавление поставок;
* Удаление поставок;
* Редактирование поставок
* Просмотр поставщиков;
* Добавление поставщиков;
* Удаление поставщиков;
* Редактирование поставщиков
* Просмотр товаров;
* Добавление товаров;
* Удаление товаров;
* Редактирование товаров;
* Просмотр сотрудников;
* Добавление сотрудников;
* Удаление сорудников;
* Редактирование сотрудников;
* Просмотр клиентов;
* Добавление клиентов;
* Удаление клиентов;
* Редактирование клиентов;
* Просмотр товара на складе;
* Поиск в категории

Основой логистики на складе является технологический процесс. Технологический процесс-это комплексное последовательное выполнение операций, связанных с подготовкой склада к приемке продукции, разгрузкой транспортных средств, приемкой продукции, хранением ее, организацией хранения, выполнением заказов, подготовкой к отпуску и транспортировкой со склада.

От того, насколько эффективно будут выполняться эти операции, будет зависеть работа других отделов (продаж, доставки, снабжения и т. д.).), Уровень логистических услуг, предоставляемых клиентам, а следовательно, и уровень конкурентоспособности компании на рынке. В связи с этим перед складским управлением каждого предприятия стоит задача оптимизации максимального количества технологических элементов.

Программа не должна иметь ограничений по разрешению и размеру дисплея пользователя (Windows должна иметь нормальный вид на 15-и 17-дюймовых дисплеях с разрешениями 1024x768 и 800x600). Программа должна работать бесперебойно на IBM-совместимом ПК, операционной системе Windows.

Чтобы пользоваться программой, пользователь должен обладать навыками работы с персональным компьютером.

В качестве примера компьютера, на котором работает программа, используется ПК со следующими характеристиками: процессор 1,9 ГГц, Оперативная память 4 ГБ, Жесткий диск 2 ТБ, монитор 17.

## 1.3 Проектирование программного продукта

В качестве предметной области в курсовой работе рассматривается склад.

Для описания всех процессов, требуется описать несколько сущностей:

* Клиент
* Поставка
* Сотрудник
* Заказ
* Товар
* Поставщик
* Склад

Сущность «Клиент» служит для отображения информации о клиентах.

Сущность «Поставка» служит для отображения информации о поставках товара на склад.

Сущность «Сотрудник» служит для отображения информации о работающих сотрудниках.

Сущность «Заказ» служит для отображения информации о заказах/отгрузке товара.

Сущность «Товар» служит для отображения информации о товарах.

Сущность «Поставщик» служит для отображения информации о поставщиках товаров.

Сущность «Склад» служит для отображения информации о товарах на складе.

Для того, чтобы понимать, к каким частям имеет доступ пользователь, построим диаграмму вариантов использования, представленную ниже на рисунке 1.1.

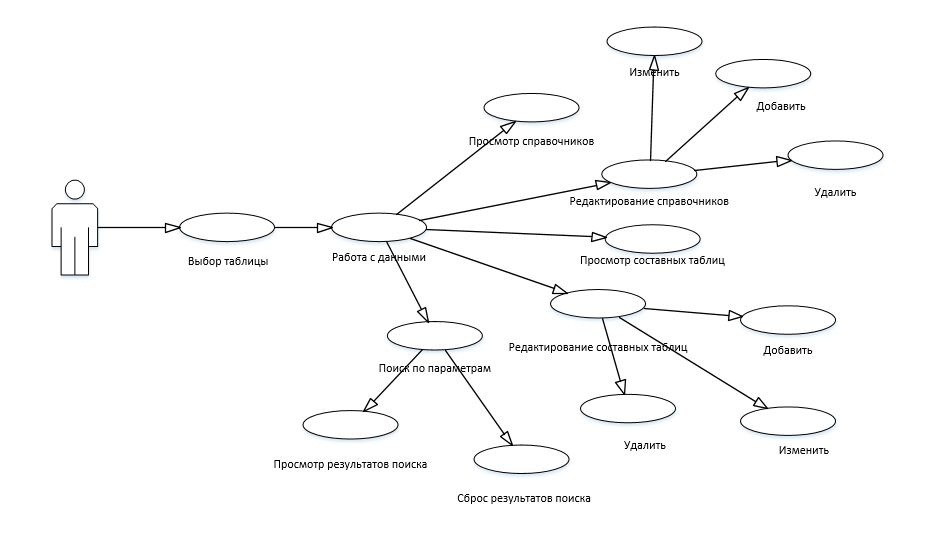


Рисунок 1.1 – Диаграмма вариантов использования

Приложение будет построено с использованием паттерна MVC [3]

* VC — это набор архитектурных идей и принципов для построения сложных информационных систем с пользовательским интерфейсом;
* MVC — это аббревиатура, которая расшифровывается так: Model-View-Controller.

MVC — это не паттерн проектирования. MVC — это именно набор архитектурных идей и принципов для построения сложных систем с пользовательским интерфейсом. Но для удобства, чтобы каждый раз не повторять: “Набор архитектурных идей…”, мы будем называть MVC паттерном. Начнем с простого. Что же скрывается за словами Model-View-Controller? При разработке систем с пользовательским интерфейсом, следуя паттерну MVC нужно разделять систему на три составные части. Их, в свою очередь, можно называть модулями или компонентами. Говори как хочешь, но дели на три. У каждой составной компоненты будет свое предназначение.

Model. Первая компонента/модуль — так называемая модель. Она содержит всю бизнес-логику приложения.

View. Вторая часть системы — вид. Данный модуль отвечает за отображение данных пользователю. Все, что видит пользователь, генерируется видом.

Controller. Третьим звеном данной цепи является контроллер. В нем хранится код, который отвечает за обработку действий пользователя (любое действие пользователя в системе обрабатывается в контроллере).

Модель — самая независимая часть системы. Настолько независимая, что она не должна ничего знать о модулях Вид и Контроллер. Модель настолько независима, что ее разработчики могут практически ничего не знать о Виде и Контроллере. Основное предназначение Вида — предоставлять информацию из Модели в удобном для восприятия пользователя формате. Основное ограничение Вида — он никак не должен изменять модель. Основное предназначение Контроллера — обрабатывать действия пользователя. Именно через Контроллер пользователь вносит изменения в модель. Точнее в данные, которые хранятся в модели.

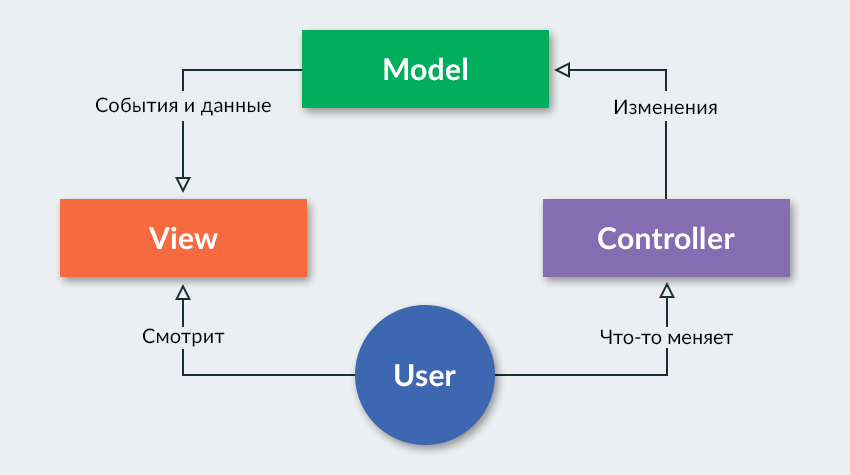


Рисунок 1.2 – взаимодействие в MVC

Из всего этого можно сделать вполне логичный вывод. Сложную систему нужно разбивать на модули.

# 2 РЕАЛИЗАЦИЯ ПРОГРАММНОГО ПРОДУКТА

## 2.1 Выбор инструментов разработки

Для выполнения задач, поставленных перед курсовой работой, были выбраны высокоуровневый язык программирования Java и платформа для написания программного кода Eclipse. Ниже приводится краткое описание языка разработки.

Язык программирования Java-это объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems с 1991 года и официально выпущенный 23 мая 1995 года [4]. Новый язык программирования, первоначально известный как Oak (Джеймс Гослинг), был разработан для бытовой электроники, но позже был переименован в Java для написания апплетов, приложений и серверного программного обеспечения.

Программы Java могут быть переведены в байт-код, который работает на виртуальной машине java (JVM), которая обрабатывает байт-код и передает инструкции аппаратному обеспечению, как интерпретатор, но разница в том, что байт-код, в отличие от текста, гораздо быстрее обрабатывается.

Язык Java родился в рамках проекта по созданию продвинутого программного обеспечения для различных бытовых приборов. Проект был начат на C++, но вскоре возникли некоторые проблемы, и лучшим способом их решения стало изменение самого инструмента–языка программирования. Очевидно, что существует потребность в независимом от платформы языке программирования, позволяющем создавать программы, которые не нужно компилировать.

Выбор системы БД лег на My SQL Server для реализации разработанной базы данных [5]. MySQL является наиболее приспособленной для применения в среде web СУБД (системой управления базами данных). Не секрет, что для исполнения приложений клиента на большинстве хостинг-площадок провайдеры предоставляют небольшое количество ресурсов (как вычислительных, так и дисковых). Поэтому для данного применения необходима высокоэффективная СУБД, обладающая при этом высокой надежностью (большинство web-приложений и сайтов должны работать в режиме 24/7).

По всем этим причинам MySQL стала незыблемым стандартом в области СУБД для web, а теперь в ней развиваются возможности для использования ее в любых критичных бизнес-приложениях, то есть конкурирует на равных с такими СУБД таких производителей, как Oracle, IBM, Microsoft и Sybase.

## 2.2 Разработка базы данных

В создаваемой базе данных будут использоваться следующие типы данных:

* INT – Целочисленный тип. Размер – 4 байта;
* VARCHAR – Строковый тип переменной длины;
* Date – тип данных который предназначен для хранения дат;

Перечень всех таблиц док представлен в таблице док 2.1.

Таблица 2.1 – Таблицы базы данных

|  |  |
| --- | --- |
| **Название** | **Содержимое** |
| Provider | Информация о поставщиках |
| Product | Информация о товарах |
| Delivery | Информация о поставках |
| Warehouse | Информация о товарах на складе |
| Employee | Информация о сотружниках |
| Client | Информация о клиентах |
| Orders | Информация о заказах (выгрузках) |

Таблица provider хранит в себе информацию о Поставщиках. Ее структура приведена в таблице 2.2.

Таблица 2.2 **– Характеристика атрибутов таблицы provider**

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| Id | Int | Идентификатор записи |
| Name | Varchar(255) | Наименование поставщика |
| Adres | Varchar(255) | Адрес поставщика |
| Phone | Varchar(255) | Телефон поставщика |
| Email | Varchar(255) | Адрес электронной почты |

Таблица product хранит в себе информацию о продуктах. Ее структура приведена в таблице 2.3.

Таблица 2.3 **– Характеристика атрибутов таблицы product**

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| Id | Int | Идентификатор записи |
| Name | Varchar(255) | Наименование продукта |
| Price | Int | Цена продукта |

Таблица delivery хранит в себе информацию о поставках. Ее структура приведена в таблице 2.4.

Таблица 2.4 **– Характеристика атрибутов таблицы delivery**

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| Id | Int | Идентификатор записи |
| Id\_product | Int | Идентификатор продукта |
| Id\_provider | Int | Идентификатор поставщика |
| Dates | date | Дата поставки |
| Count | int | Количество товара в поставке |

Таблица warehouse хранит в себе информацию о . Ее структура приведена в таблице 2.5.

Таблица 2.5 **– Характеристика атрибутов таблицы warehouse**

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| Id | Int | Идентификатор записи |
| Id\_product | Int | Идентификатор продукта |
| Count | int | Количество продукта на складе |

Таблица employee хранит в себе информацию о сторудниках. Ее структура приведена в таблице 2.6.

Таблица 2.6 **– Характеристика атрибутов таблицы employee**

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| Id | Int | Идентификатор записи |
| Fio | Varchar(255) | ФИО человека |
| Phone | Varchar(255) | Номер телефона |
| Email | Varchar(255) | Адрес электронной почты |

Таблица client хранит в себе информацию о клиентах. Ее структура приведена в таблице 2.7.

Таблица 2.7 **– Характеристика атрибутов таблицы client**

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| Id | Int | Идентификатор записи |
| Fio | Varchar(255) | ФИО человека |
| Phone | Varchar(255) | Номер телефона |
| email | Varchar(255) | Адрес электронной почты |

Таблица orders хранит в себе информацию о заказах (отгрузках). Ее структура приведена в таблице 2.8.

Таблица 2.8 **– Характеристика атрибутов таблицы orders**

|  |  |  |
| --- | --- | --- |
| Имя атрибута | Тип | Описание |
| Id | Int | Идентификатор записи |
| Id\_employee | Int | Идентификатор сотрудника |
| Id\_client | Int | Идентификатор клиента |
| Id\_product | Int | Идентификатор продукта |
| Count | Int | Количество продукта |
| dates | Date | Дата заказа |

Таким образом, имеем следующу схему базы данных

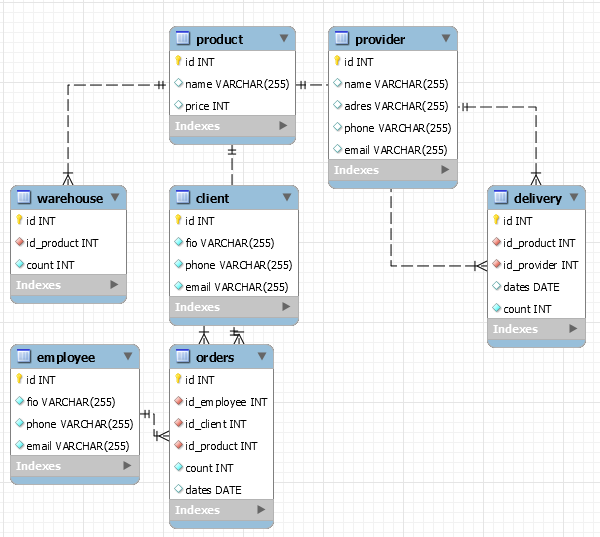


Рисунок 2.1 – схема БД

## 2.3 Описание классов программы

При написании приложения было разработано множество классов, которые разбиты на соответствующие пакеты.

Все классы из пакета forms представляют собой view JFrame и служат для отображения данных.

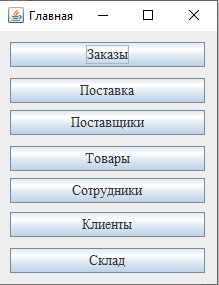


Рисунок 2.2 – Форма главного меню

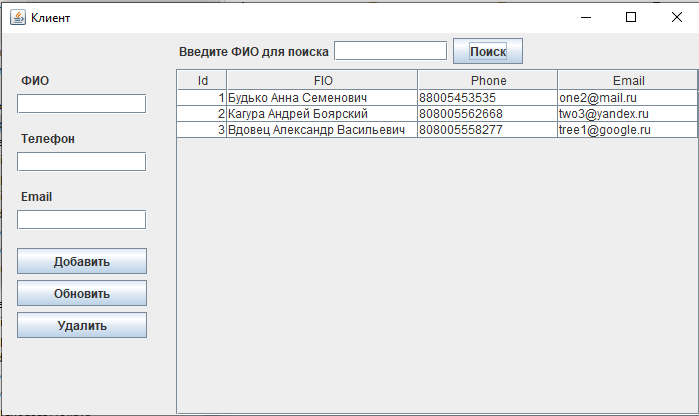


Рисунок 2.3 – Форма клиентов

К примеру, класс «ClientFrame» отвечает за отображение данных о клиентах и кнопки взаимодействия с ними. Классы «DeliveryFrame», «EmployeeFrame» и «MainFrame» отвечают за показ данны о поставках, сотрудниках и главное меню соответственно.

Поскольку работа с данными влечет за собой работу с таблицами, для каждой таблицы разработана своя модель отображения. Все эти модели хранятся в пакете «tableModal».

Так, к примеру, класс «WarehouseTableModal» отвечает за отображение данных склада в таблице, а класс «OrderTableModal» за отображение данных заказов соответственно.

Пример кода с модели для склада

**Листинг 1 – модель отображения таблицы склада**

**private** **static** **final** **int** ***ID\_COL*** = 0;

**private** **static** **final** **int** ***PRODUCT\_NAME\_COL*** = 1;

**private** **static** **final** **int** ***PRODUCT\_PRICE\_COL*** = 2;

**private** **static** **final** **int** ***COUNT\_COL*** = 3;

**private** **final** String[] columnNames = {"Id", "Product", "Price","Count" };

**private** List<Warehouse> list;

**public** WarehouseTableModal(List<Warehouse> theProviders) {

list = theProviders;

}

@Override

**public** **int** getColumnCount() {

**return** columnNames.length;

}

@Override

**public** **int** getRowCount() {

**return** list.size();

}

@Override

**public** String getColumnName(**int** col) {

**return** columnNames[col];

}

Для работ с отдельными данными, исполозуются сущности, которые находятся в пакете «entity». Примером их являются классы «Product» и «Provider», которые являются сущностями записи в таблице товаров и записью в таблице поставщик соответственно.

**Листинг 2 – сущность поставок**

**public** **class** Order **extends** AbstractEntity{

**private** Employee employee;

**private** Client client;

**private** Product product;

**private** Date date;

**private** **int** count;

**public** Order() {

**super**();

}

**public** Order(Employee employee, Client client, Product product, Date date, **int** count) {

**super**();

**this**.employee = employee;

**this**.client = client;

**this**.product = product;

**this**.date = date;

**this**.count = count;

}}

Для взаимодействия с базой данных, каждому классу необходим клас посредник. Все такие классы находятся в пакете «dao». С помощью них можно работать с данными каждой таблицы по отдельности.

**В основном работа с бд для каждой таблицы похожа. Пример добавления данных и обновления из таблицы Заказ показан ниже.**

**Листинг 3 – метод добавления и обновления данных таблицы заказ**

**public** **void** update(Order entity) **throws** Exception {

PreparedStatement myStmt = **null**;

**try** {

Order past = read(entity.getId()).get(0);

Warehouse warehouse = ApplicationController.*warehouseController*.getDAO()

.readAllByIdProduct(past.getProduct().getId()).get(0);

**int** countDifference = past.getCount() - entity.getCount();

myStmt = myConn.prepareStatement(

"UPDATE orders SET id\_employee=?, id\_client=?, id\_product=?, dates=?, count=? WHERE id=?");

myStmt.setLong(1, entity.getEmployee().getId());

myStmt.setLong(2, entity.getClient().getId());

myStmt.setLong(3, entity.getProduct().getId());

myStmt.setDate(4, entity.getDate());

myStmt.setInt(5, entity.getCount());

myStmt.setLong(6, entity.getId());

myStmt.executeUpdate();

warehouse.setCount(warehouse.getCount() + past.getCount());

ApplicationController.*warehouseController*.getDAO().update(warehouse);

List<Warehouse> readAllByIdProduct = ApplicationController.*warehouseController*.getDAO()

.readAllByIdProduct(entity.getProduct().getId());

Warehouse warehouse2 = readAllByIdProduct.get(0);

warehouse2.setCount(warehouse2.getCount() - entity.getCount());

ApplicationController.*warehouseController*.getDAO().update(warehouse2);

} **finally** {

close(myStmt);

}

}

**public** **void** create(Order entity) **throws** Exception {

PreparedStatement myStmt = **null**;

**try** {

myStmt = myConn.prepareStatement("insert into orders"

+ " (id\_employee, id\_client, id\_product, dates, count)" + " values (?, ?, ?, ?,?)");

myStmt.setLong(1, entity.getEmployee().getId());

myStmt.setLong(2, entity.getClient().getId());

myStmt.setLong(3, entity.getProduct().getId());

myStmt.setDate(4, entity.getDate());

myStmt.setInt(5, entity.getCount());

myStmt.executeUpdate();

// past

Warehouse warehouse = ApplicationController.*warehouseController*.getDAO()

.readAllByIdProduct(entity.getProduct().getId()).get(0);

**if** (warehouse.getCount() - entity.getCount() <= 0) {

ApplicationController.*warehouseController*.getDAO().Delete(warehouse.getId());

} **else** {

warehouse.setCount(warehouse.getCount() - entity.getCount());

ApplicationController.*warehouseController*.getDAO().update(warehouse);

}

} **finally** {

close(myStmt);

}

}

И для упралвения формами и данными от имени пользователя, используются контроллеры, находящиеся в пакете «controller». Каждой форме соответствует свой контроллер, который отвечает на действия пользователей. К примеру, если пользователь ввел данные на форме сотрудника и нажал кнопку добавить, все эти данные отсылаются в контроллер в соответствующий метод. Далее контроллер уже решает что с ними делать и что выводить пользователю дальше.

Пример работы контроллера клиентов показан в листинге 4

**Листинг 4 – работа контроллера клиентов**

**public** ClientController(JFrame frame) {

**super**(frame);

**try** {

DAO = **new** ClientDAO();

} **catch** (Exception e) {e.printStackTrace();}

}

**public** **void** actionSearchButton(String fio, JTable table) {

**try** {

List<Client> list = **null**;

**if** (fio!= **null** && fio.trim().length() > 0)

list = DAO.search(fio);

**else**

list = DAO.readAll();

ClientTableModal model = **new** ClientTableModal(list);

table.setModel(model);

} **catch** (Exception e) {

JOptionPane.*showMessageDialog*(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.***ERROR\_MESSAGE***);

}

}

**public** **void** actionCreateButton(String fio, String phone, String email) {

**if** (fio.length() > 0 && phone.length() > 0 && email.length() > 0) {

Client entity = **new** Client(fio, phone, email);

**try** {

DAO.create(entity);

refrechView();

JOptionPane.*showMessageDialog*(getFrame(), "Успешно добавлено", "Успех", JOptionPane.***INFORMATION\_MESSAGE***);

} **catch** (Exception e) {

JOptionPane.*showMessageDialog*(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.***ERROR\_MESSAGE***);

}

}

}

Для управления всем приложением используется класс «ApplicationController». Он содержит в себе все остальные контроллеры, и с помощью него формы могут общаться друг с другом.

Общая диаграмма классов представлена в приложении В

# 3 ТЕСТИРОВАНИЕ И РАЗВЁРТЫВАНИЕ ПРИЛОЖЕНИЯ

## 3.1 Разработка плана тестирования

Контроль качества программы-это деятельность, которая позволяет определить, соответствует ли программный продукт ожидаемым результатам (т. е. заданным требованиям) при различных условиях, входных данных и средах [6]. Качество программного продукта определяется определенными качественными характеристиками (качественная характеристика-это совокупность атрибутов программного средства, описывающая и оценивающая его качество). К качественным характеристикам первого уровня относятся:

Функция-это свойство набора программных средств, определяемое наличием и специфической функциональностью набора функций, удовлетворяющих заданному или неявному требованию. Функциональные требования определяют функциональность программного обеспечения и решаемые им задачи.

Надежность-это совокупность характеристик, которая представляет собой способность программных средств поддерживать заданный уровень пригодности в течение заданного интервала времени при заданном условии.

Доступность - это набор свойств программного средства, характеризующий усилия, необходимые для его тестирования в заданной или неявной пользовательской области. В ходе тестирования доступности было установлено, что разработанный интерфейс соответствует требованиям доступности.

Эффективность-это совокупность свойств программного средства, характеризующая те аспекты его применимости, уровень которых связан с характеристиками и временем использования ресурсов, что необходимо при заданных условиях эксплуатации.

Качество ремонтопригодности-это совокупность свойств программного средства, характеризующих работу, необходимую для его модификации.

Переносимость-это совокупность свойств программного средства, характеризующая его адаптивность к переходу из одной операционной среды в другую.

В зависимости от направления теста проверьте конкретную функциональность приложения. Как правило, процесс тестирования записывается в виде плана тестирования и тестового случая (test case). План тестирования описывает стратегию тестирования,методы и инструменты тестирования, процесс тестирования и другие функции. TestCase описывает последовательную пошаговую операцию,используемую для проверки функциональности программных средств. Это минимальные базовые операции выверки для каждой функции или элемента приложения.

## 3.2 Тестирование приложение

Программа тестирования включает в себя проверку его работы в различных условиях для определения правильности его работы.

Разработка любой программы предполагает наличие ошибок в исходном тексте и борьбу с ними. Все почти бесчисленные возможные ошибки обычно делятся на несколько групп:

Синтаксическая ошибка;

Семантическая (логическая) ошибка.

Синтаксические ошибки-это самые простые ошибки, которые легко исправить на этапе компиляции. Их причина заключается в том, что в сервисе, значит, один из операторов набрал неверный ввод.

Семантические (логические) ошибки-самые сложные и неуловимые. Они ведут себя так, что программа действует не так, как ожидалось. Последствия семантических ошибок могут быть самыми разными: неправильное содержимое окна, невыполненное или неправильное выполнение пользовательских команд, неправильное содержимое таблицы и так далее. Обработка семантических ошибок требует почти всего времени отладки.

В таблице 3.1 приведены результаты тестирования программного обеспечения. Используйте метод анализа граничных условий для тестирования.

**Таблица 3.1 – Test Case**

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Описание Test Case | Ожидаемый  результат | Полученный  Результат |
| 1 | 1.Открыть программу  2.Перейти к списку заказов | Открывается форма заказов | Открывается форма заказов |
| 2 | 1.Открыть программу  2.Перейти к списку заказов.  3.Добавить новый заказ, указав продукт ручка, количество продукта 2. | В таблицу добавился новый заказ. В таблице склада из количество товара ручка вычтено 2 товара | В таблице отображен новый заказ. В таблице склада количество товаров ручка уменьшилось на 2 |
| 3 | 1.Открыть программу  2.Перейти к списку заказов.  3.Добавить новый заказ, указав продукт ручка, количество продукта больше чем есть на складе | Показ пользователю соощения об ошибке | Пользователю показалось уведомление об ошибке и пояснение, что он пытается взять больше товара, чем есть на складе |
| 4 | 1.Открыть программу  2.Перейти к списку поставок.  3.Обновить существующую поставку, указав количество продукта на 3 больше, чем было | Обновились данные о поставках с указанным новым количеством, в таблице склад количество выбранного товара увеличилась на разность количества до обновления и нового значения | Обновились данные о поставках с указанным новым количеством, в таблице склад количество выбранного товара увеличилась на разность количества до обновления и нового значения |

## 3.3 Развертывание приложения

Для запуска приложения у клиента должно быть установлено следующее программное обеспечение:

* Виртуальная машина Java и Jdk 11;
* Среда разработки Eclipse;
* My Sql Server;

После того как пользователь установил данное программное обеспечение следует импортировать создать базу данных, использовав скрипт «createDB» и открыть проект в IDE.

Для запуска приложения можно использовать комбинацию клавиш Alt+shift+x или круглую иконку с зеленым фоном и белым знаком воспроизведения.

Для экспортирования проекта как приложение, нужны выбрать проект из списка проектов, нажать по нему правой кнопкой мыши. Из перечисленных вариантов выбрать Export-java-Runnable JAR file. Далее выбрать место для хранения и нажать кнопку Export. После чего запускать приложение можно без открытия IDE.

# ЗАКЛЮЧЕНИЕ

Результатом курсового проекта стала разработка программного обеспечения для автоматизации учета товаров на складах.

Во время написания курсового проекта я подробно изучил функциональность и возможности объектно-ориентированного языка Java и приобрел навыки рационального использования современного программного обеспечения персональных компьютеров для решения различных задач.

В результате курсового проекта все цели и задачи, поставленные для его разработки на начальном этапе, были полностью реализованы.

Программа соответствует исходному техническому заданию. В теории, ее можно использовать для большинства работ, связанных с хранением товара не только на складе, но и в магазинах и похожих на их местах

По итогам всех испытаний можно смело сделать вывод, что программный продукт соответствует всем заданным критериям оценки, а также его высокой надежности и простоте использования.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Иванов, А.А. Автоматизация технологических процессов и производств: Учебное пособие / А.А. Иванов. - М.: Форум, 2012. - 224 c.
2. Клюев, А.С. Наладка средств автоматизации и автоматических систем регулирования: Справочное пособие / А.С. Клюев, А.Т. Лебедев, С.А. Клюев. - М.: Альянс, 2009. - 368 c
3. MVC [Электронный ресурс]. – Режим доступа: https://javarush.ru/groups/posts/2536-chastjh-7-znakomstvo-s-patternom-mvc-model-view-controller. – Дата доступа: 10.10.2020.
4. Гупта, Арун Java EE 7. Основы / Арун Гупта. - М. : Вильямс, 2014. - 336 c
5. MySql [Электронный ресурс]. – Режим доступа: https://www.methodlab.ru/technology/mysql.shtml. – Дата доступа: 10.10.2020.
6. Куликов, С. C. / Тестирование программного обеспечения. Базовый курс / С. С. Куликов. — Минск: Четыре четверти, 2017. — 312 с

# ПРИЛОЖЕНИЕ А. КОД ПРОГРАММЫ

package tableModal;

import java.util.List;

import javax.swing.table.AbstractTableModel;

import entity.Client;

import entity.Warehouse;

/\*\*

\*

\* @author admin

\*Модель таблицы склада

\*/

public class WarehouseTableModal extends AbstractTableModel {

private static final int ID\_COL = 0;

private static final int PRODUCT\_NAME\_COL = 1;

private static final int PRODUCT\_PRICE\_COL = 2;

private static final int COUNT\_COL = 3;

private final String[] columnNames = {"Id", "Product", "Price","Count" };

private List<Warehouse> list;

public WarehouseTableModal(List<Warehouse> theProviders) {

list = theProviders;

}

@Override

public int getColumnCount() {

return columnNames.length;

}

@Override

public int getRowCount() {

return list.size();

}

@Override

public String getColumnName(int col) {

return columnNames[col];

}

@Override

public Object getValueAt(int row, int col) {

Warehouse temp = list.get(row);

switch (col) {

case ID\_COL:

return temp.getId();

case PRODUCT\_NAME\_COL:

return temp.getProduct().getName();

case PRODUCT\_PRICE\_COL:

return temp.getProduct().getPrice();

case COUNT\_COL:

return temp.getCount();

default:

return temp;

}

}

@Override

public Class getColumnClass(int c) {

return getValueAt(0, c).getClass();

}

}

package tableModal;

import java.util.List;

import javax.swing.table.AbstractTableModel;

import entity.Provider;

/\*\*

\*

\* @author admin

\* Модель таблицы поставщика

\*/

public class ProviderTableModal extends AbstractTableModel {

private static final int ID\_COL = 0;

private static final int NAME\_COL = 1;

private static final int ADRES\_COL = 2;

private static final int PHONE\_COL = 3;

private static final int EMAIL\_COL = 4;

private final String[] columnNames = {"Id", "Name", "Adres", "Phone", "Email" };

private List<Provider> list;

public ProviderTableModal(List<Provider> theProviders) {

list = theProviders;

}

@Override

public int getColumnCount() {

return columnNames.length;

}

@Override

public int getRowCount() {

return list.size();

}

@Override

public String getColumnName(int col) {

return columnNames[col];

}

@Override

public Object getValueAt(int row, int col) {

Provider temp = list.get(row);

switch (col) {

case ID\_COL:

return temp.getId();

case NAME\_COL:

return temp.getName();

case ADRES\_COL:

return temp.getAdres();

case PHONE\_COL:

return temp.getPhone();

case EMAIL\_COL:

return temp.getEmail();

default:

return temp.getName();

}

}

@Override

public Class getColumnClass(int c) {

return getValueAt(0, c).getClass();

}

}

package tableModal;

import java.util.List;

import javax.swing.table.AbstractTableModel;

import entity.Product;

/\*\*

\*

\* @author admin

\* модель таблицы товара

\*/

public class ProductTableModal extends AbstractTableModel {

private static final int ID\_COL = 0;

private static final int NAME\_COL = 1;

private static final int PRICE\_COL = 2;

private final String[] columnNames = {"Id", "Name", "Price"};

private List<Product > list;

public ProductTableModal(List<Product > theProviders) {

list = theProviders;

}

@Override

public int getColumnCount() {

return columnNames.length;

}

@Override

public int getRowCount() {

return list.size();

}

@Override

public String getColumnName(int col) {

return columnNames[col];

}

@Override

public Object getValueAt(int row, int col) {

Product temp = list.get(row);

switch (col) {

case ID\_COL:

return temp.getId();

case NAME\_COL:

return temp.getName();

case PRICE\_COL:

return temp.getPrice();

default:

return temp.getName();

}

}

@Override

public Class getColumnClass(int c) {

return getValueAt(0, c).getClass();

}

}

package tableModal;

import java.util.List;

import javax.swing.table.AbstractTableModel;

import entity.Order;

/\*\*

\*

\* @author admin

\* модель таблицы заказа

\*/

public class OrderTableModal extends AbstractTableModel {

public static final int ID\_COL = 0;

public static final int EMPLOYEE\_COL = 1;

public static final int CLIENT\_COL = 2;

public static final int PRODUCT\_COL = 3;

public static final int COUNT\_COL = 4;

public static final int DATES\_COL = 5;

public static final int ENTITY\_COL = 6;

private final String[] columnNames = {"Id","Employee", "Client", "Product", "Count","Date" };

private List<Order> list;

public OrderTableModal(List<Order> theProviders) {

list = theProviders;

}

@Override

public int getColumnCount() {

return columnNames.length;

}

@Override

public int getRowCount() {

return list.size();

}

@Override

public String getColumnName(int col) {

return columnNames[col];

}

@Override

public Object getValueAt(int row, int col) {

Order temp = list.get(row);

switch (col) {

case ID\_COL:

return temp.getId();

case EMPLOYEE\_COL:

return temp.getEmployee().getFio();

case CLIENT\_COL:

return temp.getClient().getFio();

case PRODUCT\_COL:

return temp.getProduct().getName();

case COUNT\_COL:

return temp.getCount();

case DATES\_COL:

return temp.getDate();

default:

return temp;

}

}

@Override

public Class getColumnClass(int c) {

return getValueAt(0, c).getClass();

}

}

package tableModal;

import java.util.List;

import javax.swing.table.AbstractTableModel;

import entity.Employee;

/\*\*

\*

\* @author admin

\* Модель таблицы сотрудника

\*/

public class EmployeeTableModal extends AbstractTableModel {

private static final int ID\_COL = 0;

private static final int FIO\_COL = 1;

private static final int PHONE\_COL = 2;

private static final int EMAIL\_COL = 3;

private final String[] columnNames = {"Id", "FIO", "Phone", "Email" };

private List<Employee > list;

public EmployeeTableModal(List<Employee > theProviders) {

list = theProviders;

}

@Override

public int getColumnCount() {

return columnNames.length;

}

@Override

public int getRowCount() {

return list.size();

}

@Override

public String getColumnName(int col) {

return columnNames[col];

}

@Override

public Object getValueAt(int row, int col) {

Employee temp = list.get(row);

switch (col) {

case ID\_COL:

return temp.getId();

case FIO\_COL:

return temp.getFio();

case PHONE\_COL:

return temp.getPhone();

case EMAIL\_COL:

return temp.getEmail();

default:

return temp.getFio();

}

}

@Override

public Class getColumnClass(int c) {

return getValueAt(0, c).getClass();

}

}

package tableModal;

import java.util.List;

import javax.swing.table.AbstractTableModel;

import entity.Client;

import entity.Delivery;

/\*\*

\* модель таблицы поставок

\* @author admin

\*

\*/

public class DeliveryTableModal extends AbstractTableModel {

public static final int ID\_COL = 0;

public static final int PRODUCT\_COL = 1;

public static final int PROVIDER\_COL = 2;

public static final int DATE\_COL = 3;

public static final int COUNT\_COL = 4;

public static final int ENTITY\_COL = 5;

private final String[] columnNames = { "Id", "Product", "Provider", "Date", "Count" };

private List<Delivery> list;

public DeliveryTableModal(List<Delivery> theProviders) {

list = theProviders;

}

@Override

public int getColumnCount() {

return columnNames.length;

}

@Override

public int getRowCount() {

return list.size();

}

@Override

public String getColumnName(int col) {

return columnNames[col];

}

@Override

public Object getValueAt(int row, int col) {

Delivery temp = list.get(row);

switch (col) {

case ID\_COL:

return temp.getId();

case PRODUCT\_COL:

return temp.getProduct().getName();

case PROVIDER\_COL:

return temp.getProvider().getName();

case DATE\_COL:

return temp.getDate();

case COUNT\_COL:

return temp.getCount();

case ENTITY\_COL:

return temp;

default:

return temp.getProduct().getName() + " " + temp.getCount();

}

}

@Override

public Class getColumnClass(int c) {

return getValueAt(0, c).getClass();

}

}

package tableModal;

import java.util.List;

import javax.swing.table.AbstractTableModel;

import entity.Client;

/\*\*

\* модель таблицы клиентов

\* @author admin

\*

\*/

public class ClientTableModal extends AbstractTableModel {

private static final int ID\_COL = 0;

private static final int FIO\_COL = 1;

private static final int PHONE\_COL = 2;

private static final int EMAIL\_COL = 3;

private final String[] columnNames = {"Id", "FIO", "Phone", "Email" };

private List<Client> list;

public ClientTableModal(List<Client> theProviders) {

list = theProviders;

}

@Override

public int getColumnCount() {

return columnNames.length;

}

@Override

public int getRowCount() {

return list.size();

}

@Override

public String getColumnName(int col) {

return columnNames[col];

}

@Override

public Object getValueAt(int row, int col) {

Client temp = list.get(row);

switch (col) {

case ID\_COL:

return temp.getId();

case FIO\_COL:

return temp.getFio();

case PHONE\_COL:

return temp.getPhone();

case EMAIL\_COL:

return temp.getEmail();

default:

return temp.getFio();

}

}

@Override

public Class getColumnClass(int c) {

return getValueAt(0, c).getClass();

}

}

package forms;

import java.awt.BorderLayout;

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import javax.swing.GroupLayout;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.GroupLayout.Alignment;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.border.EmptyBorder;

import controller.ApplicationController;

/\*\*

\* форма отображения склада

\* @author admin

\*

\*/

public class WarehouseFrame extends JFrame {

private JTextField textField;

private JTable table;

public WarehouseFrame() {

setTitle("Склад");

setBounds(100, 100, 592, 428);

getContentPane().setLayout(new BorderLayout(0, 0));

JPanel panel = new JPanel();

getContentPane().add(panel, BorderLayout.NORTH);

JLabel lblNewLabel\_4 = new JLabel("Введите название продукта для поиска");

panel.add(lblNewLabel\_4);

textField = new JTextField();

textField.setColumns(10);

panel.add(textField);

JButton btnNewButton\_1 = new JButton("Поиск");

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionSearchButton();

}

});

panel.add(btnNewButton\_1);

table = new JTable();

getContentPane().add(new JScrollPane(table), BorderLayout.CENTER);

}

private void actionSearchButton() {

ApplicationController.warehouseController.actionSearchButton(textField.getText().trim(), table);

}

public void refreshView() {

ApplicationController.warehouseController.actionSearchButton("", table);

}

}

package forms;

import java.awt.BorderLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.border.EmptyBorder;

import controller.ApplicationController;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

/\*\*

\* форма отображения поставщиков

\* @author admin

\*

\*/

public class ProviderFrame extends JFrame {

private JTable table;

private JTextField textField;

private JTextField textField\_1;

private JTextField textField\_2;

private JTextField textField\_3;

private JTextField textField\_4;

public ProviderFrame() {

setTitle("Поставщик");

setDefaultCloseOperation(JFrame.HIDE\_ON\_CLOSE);

setBounds(100, 100, 592, 428);

getContentPane().setLayout(new BorderLayout());

JPanel panel = new JPanel();

getContentPane().add(panel, BorderLayout.NORTH);

JLabel lblNewLabel\_4 = new JLabel("Введите имя для поиска");

panel.add(lblNewLabel\_4);

textField\_4 = new JTextField();

panel.add(textField\_4);

textField\_4.setColumns(10);

JButton btnNewButton\_1 = new JButton("Поиск");

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionSearchButton();

}

});

panel.add(btnNewButton\_1);

JPanel panel\_1 = new JPanel();

panel\_1.setBorder(new EmptyBorder(3, 3, 3, 3));

getContentPane().add(panel\_1, BorderLayout.WEST);

JLabel lblNewLabel = new JLabel("Имя");

textField = new JTextField();

textField.setColumns(10);

JLabel lblNewLabel\_1 = new JLabel("Адрес");

textField\_1 = new JTextField();

textField\_1.setColumns(10);

JLabel lblNewLabel\_2 = new JLabel("Телефон");

textField\_2 = new JTextField();

textField\_2.setColumns(10);

JLabel lblNewLabel\_3 = new JLabel("Почта");

textField\_3 = new JTextField();

textField\_3.setColumns(10);

JButton btnNewButton = new JButton("Добавить");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionCreateButton();

}

});

JButton btnNewButton\_2 = new JButton("Обновить");

btnNewButton\_2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionUpdateButton();

}

});

JButton btnNewButton\_3 = new JButton("Удалить");

btnNewButton\_3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionDeleteButton();

}

});

GroupLayout gl\_panel\_1 = new GroupLayout(panel\_1);

gl\_panel\_1.setHorizontalGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup().addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup().addGap(4).addComponent(lblNewLabel))

.addGroup(gl\_panel\_1.createSequentialGroup().addGap(4).addComponent(lblNewLabel\_1))

.addGroup(gl\_panel\_1.createSequentialGroup().addGap(4).addComponent(lblNewLabel\_2))

.addGroup(gl\_panel\_1.createSequentialGroup().addGap(4).addComponent(lblNewLabel\_3))

.addComponent(textField, GroupLayout.PREFERRED\_SIZE, 130, GroupLayout.PREFERRED\_SIZE)

.addComponent(textField\_1, GroupLayout.PREFERRED\_SIZE, 130, GroupLayout.PREFERRED\_SIZE)

.addComponent(textField\_2, GroupLayout.PREFERRED\_SIZE, 130, GroupLayout.PREFERRED\_SIZE)

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.TRAILING, false)

.addComponent(btnNewButton\_2, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE,

GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(btnNewButton, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE,

GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(textField\_3, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE, 130,

Short.MAX\_VALUE))

.addComponent(btnNewButton\_3, GroupLayout.PREFERRED\_SIZE, 130, GroupLayout.PREFERRED\_SIZE))

.addContainerGap(20, Short.MAX\_VALUE)));

gl\_panel\_1.setVerticalGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup().addComponent(lblNewLabel)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(textField, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addGap(16).addComponent(lblNewLabel\_1).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(textField\_1, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addGap(16).addComponent(lblNewLabel\_2).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(textField\_2, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addGap(16).addComponent(lblNewLabel\_3).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(textField\_3, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addGap(22).addComponent(btnNewButton).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(btnNewButton\_2).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(btnNewButton\_3).addGap(5)));

panel\_1.setLayout(gl\_panel\_1);

table = new JTable();

JScrollPane scrollPane = new JScrollPane(table);

table.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseReleased(java.awt.event.MouseEvent evt) {

actionTableMouseClicked();

}

});

getContentPane().add(scrollPane, BorderLayout.CENTER);

}

private void actionDeleteButton() {

if(table.getRowCount()>0 && table.getSelectedRowCount()>0) {

int row = table.getSelectedRow();

String id=table.getModel().getValueAt(row, 0).toString();

ApplicationController.providerController.actionDeleteButton(Long.parseLong(id));

}

}

private void actionTableMouseClicked() {

if (table.getRowCount() > 0)

try {

int row = table.getSelectedRow();

textField.setText(table.getModel().getValueAt(row, 1).toString());

textField\_1.setText(table.getModel().getValueAt(row, 2).toString());

textField\_2.setText(table.getModel().getValueAt(row, 3).toString());

textField\_3.setText(table.getModel().getValueAt(row, 4).toString());

} catch (Exception e) {

}

}

private void actionUpdateButton() {

int column = 0;

int row = table.getSelectedRow();

Long id = Long.parseLong(table.getModel().getValueAt(row, column).toString());

ApplicationController.providerController.actionUpdateButton(textField.getText().trim(),

textField\_1.getText().trim(), textField\_2.getText().trim(), textField\_3.getText().trim(), id);

}

private void actionCreateButton() {

ApplicationController.providerController.actionCreateButton(textField.getText().trim(),

textField\_1.getText().trim(), textField\_2.getText().trim(), textField\_3.getText().trim());

}

private void actionSearchButton() {

// textField\_4

ApplicationController.providerController.actionSearchButton(textField\_4.getText().trim(), table);

}

public void refreshView() {

ApplicationController.providerController.actionSearchButton("", table);

}

}

package forms;

import java.awt.BorderLayout;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import controller.ApplicationController;

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.JButton;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.JScrollPane;

import java.awt.Component;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.ActionEvent;

import javax.swing.JTable;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

/\*\*

\* форма отображения продуктов

\* @author admin

\*

\*/

public class ProductFrame extends JFrame {

private JTextField textField;

private JTextField textField\_1;

private JTextField textField\_2;

private JTable table;

public ProductFrame() {

setTitle("Товар");

setBounds(100, 100, 592, 428);

getContentPane().setLayout(new BorderLayout(0, 0));

JPanel panel = new JPanel();

getContentPane().add(panel, BorderLayout.NORTH);

JLabel lblNewLabel\_4 = new JLabel("Введите название для поиска");

panel.add(lblNewLabel\_4);

textField = new JTextField();

textField.setColumns(10);

panel.add(textField);

JButton btnNewButton\_1 = new JButton("Поиск");

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionSearchButton();

}

});

panel.add(btnNewButton\_1);

JPanel panel\_1 = new JPanel();

panel\_1.setBorder(new EmptyBorder(3, 3, 3, 3));

getContentPane().add(panel\_1, BorderLayout.WEST);

JLabel lblNewLabel = new JLabel("Название");

JLabel lblNewLabel\_1 = new JLabel("Цена");

textField\_1 = new JTextField();

textField\_1.setColumns(10);

textField\_2 = new JTextField();

textField\_2.setColumns(10);

textField\_2.addKeyListener(new KeyListener() {

@Override

public void keyTyped(KeyEvent keyEvent) {

char c = keyEvent.getKeyChar();

if (!(Character.isDigit(c)) || c == KeyEvent.VK\_BACK\_SPACE || c == KeyEvent.VK\_DELETE) {

keyEvent.consume();

}

}

@Override

public void keyReleased(KeyEvent keyEvent) {

}

@Override

public void keyPressed(KeyEvent e) {

// TODO Auto-generated method stub

}

});

JButton btnNewButton\_2 = new JButton("Обновить");

btnNewButton\_2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionUpdateButton();

}

});

JButton btnNewButton = new JButton("Добавить");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionCreateButton();

}

});

JButton btnNewButton\_3 = new JButton("Удалить");

btnNewButton\_3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionDeleteButton();

}

});

GroupLayout gl\_panel\_1 = new GroupLayout(panel\_1);

gl\_panel\_1.setHorizontalGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup().addContainerGap()

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.TRAILING)

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING, false)

.addComponent(btnNewButton\_3, GroupLayout.DEFAULT\_SIZE, 130, Short.MAX\_VALUE)

.addComponent(

btnNewButton, GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE,

Short.MAX\_VALUE)

.addComponent(btnNewButton\_2, GroupLayout.DEFAULT\_SIZE,

GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING, false)

.addGroup(

gl\_panel\_1.createSequentialGroup().addGap(4).addComponent(lblNewLabel))

.addGroup(gl\_panel\_1.createSequentialGroup().addGap(4)

.addComponent(lblNewLabel\_1))

.addComponent(textField\_1, GroupLayout.DEFAULT\_SIZE, 130, Short.MAX\_VALUE)

.addComponent(textField\_2, GroupLayout.DEFAULT\_SIZE, 130, Short.MAX\_VALUE)))

.addContainerGap(26, Short.MAX\_VALUE)));

gl\_panel\_1.setVerticalGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup().addComponent(lblNewLabel)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(textField\_1, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addGap(16).addComponent(lblNewLabel\_1).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(textField\_2, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addGap(16).addComponent(btnNewButton).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(btnNewButton\_2).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(btnNewButton\_3).addGap(233)));

panel\_1.setLayout(gl\_panel\_1);

table = new JTable();

table.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

actionTableMouseClicked();

}

});

getContentPane().add(new JScrollPane(table), BorderLayout.CENTER);

}

private void actionDeleteButton() {

if (table.getRowCount() > 0 && table.getSelectedRowCount() > 0) {

int row = table.getSelectedRow();

String id = table.getModel().getValueAt(row, 0).toString();

ApplicationController.productController.actionDeleteButton(Long.parseLong(id));

}

}

private void actionTableMouseClicked() {

if (table.getRowCount() > 0)

try {

int row = table.getSelectedRow();

textField\_1.setText(table.getModel().getValueAt(row, 1).toString());

textField\_2.setText(table.getModel().getValueAt(row, 2).toString());

} catch (Exception e) {

}

}

private void actionUpdateButton() {

int column = 0;

int row = table.getSelectedRow();

Long id = Long.parseLong(table.getModel().getValueAt(row, column).toString());

ApplicationController.productController.actionUpdateButton(textField\_1.getText().trim(),

Integer.parseInt(textField\_2.getText().trim()), id);

}

private void actionCreateButton() {

ApplicationController.productController.actionCreateButton(textField\_1.getText().trim(),

Integer.parseInt(textField\_2.getText().trim()));

}

private void actionSearchButton() {

ApplicationController.productController.actionSearchButton(textField.getText().trim(), table);

}

public void refreshView() {

ApplicationController.productController.actionSearchButton("", table);

}

}

package forms;

import java.awt.BorderLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.sql.Date;

import javax.swing.DefaultComboBoxModel;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.border.EmptyBorder;

import com.toedter.calendar.JDateChooser;

import controller.ApplicationController;

import entity.Client;

import entity.Employee;

import entity.Order;

import entity.Product;

import entity.Warehouse;

import tableModal.OrderTableModal;

/\*\*

\* форма отображения заказов

\* @author admin

\*

\*/

public class OrderFrame extends JFrame {

private Long id\_product, id\_employee, id\_client;

private JTextField textField;

private JTextField textField\_1;

private JTable table;

private JDateChooser dateChooser;

JComboBox<Employee> comboBox;

JComboBox<Client> comboBox\_1;

JComboBox<Warehouse> comboBox\_2;

public OrderFrame() {

setTitle("Заказы");

setBounds(100, 100, 810, 445);

getContentPane().setLayout(new BorderLayout(0, 0));

JPanel panel = new JPanel();

getContentPane().add(panel, BorderLayout.NORTH);

JLabel lblNewLabel\_4 = new JLabel("Введите название продукта для поиска");

panel.add(lblNewLabel\_4);

textField = new JTextField();

textField.setColumns(10);

panel.add(textField);

JButton btnNewButton\_1 = new JButton("Поиск");

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionSearchButton();

}

});

panel.add(btnNewButton\_1);

JPanel panel\_1 = new JPanel();

panel\_1.setBorder(new EmptyBorder(3, 3, 3, 3));

getContentPane().add(panel\_1, BorderLayout.WEST);

JLabel lblNewLabel = new JLabel("Сотрудник");

JLabel lblNewLabel\_1 = new JLabel("Клиент");

JLabel lblNewLabel\_2 = new JLabel("Дата");

textField\_1 = new JTextField();

textField\_1.setColumns(10);

textField\_1.addKeyListener(new KeyListener() {

@Override

public void keyTyped(KeyEvent keyEvent) {

char c = keyEvent.getKeyChar();

if (!(Character.isDigit(c)) || c == KeyEvent.VK\_BACK\_SPACE || c == KeyEvent.VK\_DELETE) {

keyEvent.consume();

}

}

@Override

public void keyReleased(KeyEvent keyEvent) {

}

@Override

public void keyPressed(KeyEvent e) {

// TODO Auto-generated method stub

}

});

JButton btnNewButton\_2 = new JButton("Обновить");

btnNewButton\_2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionUpdateButton();

}

});

JButton btnNewButton = new JButton("Добавить");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionCreateButton();

}

});

JButton btnNewButton\_3 = new JButton("Удалить");

btnNewButton\_3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionDeleteButton();

}

});

JLabel lblNewLabel\_2\_1 = new JLabel("Количество");

dateChooser = new JDateChooser();

dateChooser.setDateFormatString("yyyy-MM-dd");

comboBox = new JComboBox();

comboBox\_1 = new JComboBox();

JLabel lblNewLabel\_1\_1 = new JLabel("Продукт");

comboBox\_2 = new JComboBox();

GroupLayout gl\_panel\_1 = new GroupLayout(panel\_1);

gl\_panel\_1.setHorizontalGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING).addGroup(gl\_panel\_1

.createSequentialGroup().addContainerGap()

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(Alignment.TRAILING,

gl\_panel\_1.createSequentialGroup().addGap(4).addComponent(lblNewLabel)

.addContainerGap(108, Short.MAX\_VALUE))

.addGroup(Alignment.TRAILING, gl\_panel\_1.createSequentialGroup()

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING, false)

.addComponent(btnNewButton\_3, GroupLayout.DEFAULT\_SIZE, 130, Short.MAX\_VALUE)

.addComponent(btnNewButton, GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE,

Short.MAX\_VALUE)

.addComponent(btnNewButton\_2, GroupLayout.DEFAULT\_SIZE,

GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addGap(26))

.addGroup(Alignment.TRAILING,

gl\_panel\_1.createSequentialGroup().addComponent(lblNewLabel\_1).addContainerGap(131,

Short.MAX\_VALUE))

.addGroup(Alignment.TRAILING,

gl\_panel\_1.createSequentialGroup()

.addComponent(lblNewLabel\_1\_1, GroupLayout.PREFERRED\_SIZE, 57,

GroupLayout.PREFERRED\_SIZE)

.addContainerGap(111, Short.MAX\_VALUE))

.addGroup(Alignment.TRAILING,

gl\_panel\_1.createSequentialGroup().addComponent(lblNewLabel\_2\_1).addContainerGap(108,

Short.MAX\_VALUE))

.addGroup(Alignment.TRAILING,

gl\_panel\_1.createSequentialGroup().addComponent(lblNewLabel\_2).addContainerGap(142,

Short.MAX\_VALUE))

.addGroup(Alignment.TRAILING, gl\_panel\_1.createSequentialGroup()

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.TRAILING)

.addComponent(comboBox, Alignment.LEADING, 0, 142, Short.MAX\_VALUE)

.addComponent(comboBox\_1, Alignment.LEADING, 0, 142, Short.MAX\_VALUE)

.addComponent(dateChooser, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE, 142,

Short.MAX\_VALUE)

.addComponent(textField\_1, GroupLayout.DEFAULT\_SIZE, 142, Short.MAX\_VALUE))

.addGap(26))

.addGroup(gl\_panel\_1.createSequentialGroup().addComponent(comboBox\_2, 0, 142, Short.MAX\_VALUE)

.addGap(26)))));

gl\_panel\_1.setVerticalGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING).addGroup(gl\_panel\_1

.createSequentialGroup().addComponent(lblNewLabel).addGap(2)

.addComponent(comboBox, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(ComponentPlacement.UNRELATED).addComponent(lblNewLabel\_1)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(comboBox\_1, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(ComponentPlacement.UNRELATED).addComponent(lblNewLabel\_1\_1)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(comboBox\_2, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(ComponentPlacement.RELATED, 12, Short.MAX\_VALUE).addComponent(lblNewLabel\_2)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(dateChooser, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addGap(11).addComponent(lblNewLabel\_2\_1).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(textField\_1, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(ComponentPlacement.UNRELATED).addComponent(btnNewButton)

.addPreferredGap(ComponentPlacement.RELATED).addComponent(btnNewButton\_2)

.addPreferredGap(ComponentPlacement.RELATED).addComponent(btnNewButton\_3).addContainerGap()));

panel\_1.setLayout(gl\_panel\_1);

table = new JTable();

table.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

actionTableMouseClicked();

}

});

getContentPane().add(new JScrollPane(table), BorderLayout.CENTER);

dateChooser.setDate(new java.util.Date());

}

private void actionDeleteButton() {

if (table.getRowCount() > 0 && table.getSelectedRowCount() > 0) {

int row = table.getSelectedRow();

String id = table.getModel().getValueAt(row, 0).toString();

ApplicationController.orderController.actionDeleteButton(Long.parseLong(id));

}

}

private void actionTableMouseClicked() {

if (table.getRowCount() > 0)

try {

int row = table.getSelectedRow();

Order order = (Order) table.getModel().getValueAt(row, OrderTableModal.ENTITY\_COL);

textField\_1.setText(order.getCount() + "");

Date date = Date.valueOf(order.getDate().toString());

dateChooser.setDate(date);

Warehouse warehouse = ApplicationController.warehouseController.getDAO()

.readAllByIdProduct(order.getProduct().getId()).get(0);

comboBox.getModel().setSelectedItem(order.getEmployee());

comboBox\_1.getModel().setSelectedItem(order.getClient());

comboBox\_2.getModel().setSelectedItem(warehouse);

} catch (Exception e) {

}

}

private void actionUpdateButton() {

Warehouse warehouse = ((Warehouse) (comboBox\_2.getSelectedItem()));

Order order = (Order) table.getModel().getValueAt(table.getSelectedRow(), OrderTableModal.ENTITY\_COL);

Long id = order.getId();

id\_employee = ((Employee) (comboBox.getSelectedItem())).getId();

id\_client = ((Client) (comboBox\_1.getSelectedItem())).getId();

id\_product = warehouse.getProduct().getId();

int count = Integer.parseInt(textField\_1.getText());

int countWarehouse = warehouse.getCount();

int countOrder = order.getCount();

if (id\_product == order.getProduct().getId()) {

if (count > (countWarehouse + countOrder)) {

JOptionPane.showMessageDialog(this, "Вы пытаетесь заказать больше товара, чем есть на складе.\n"

+ "Всего на складке " + countWarehouse, "Ошибка", JOptionPane.ERROR\_MESSAGE);

return;

}

} else if (count > countWarehouse) {

JOptionPane.showMessageDialog(this, "Вы пытаетесь заказать больше товара, чем есть на складе.\n"

+ "Всего на складке " + warehouse.getCount(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

return;

}

ApplicationController.orderController.actionUpdateButton(id\_employee, id\_client, id\_product,

new Date(dateChooser.getDate().getTime()), Integer.parseInt(textField\_1.getText()), id);

}

private void actionCreateButton() {

Warehouse warehouse = ((Warehouse) (comboBox\_2.getSelectedItem()));

id\_employee = ((Employee) (comboBox.getSelectedItem())).getId();

id\_client = ((Client) (comboBox\_1.getSelectedItem())).getId();

id\_product = warehouse.getProduct().getId();

int count = Integer.parseInt(textField\_1.getText());

int countPast = warehouse.getCount();

if (count > countPast) {

JOptionPane.showMessageDialog(this, "Вы пытаетесь заказать больше товара, чем есть на складе.\n"

+ "Всего на складке " + warehouse.getCount(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

return;

}

ApplicationController.orderController.actionCreateButton(id\_employee, id\_client, id\_product,

new Date(dateChooser.getDate().getTime()), count);

}

private void actionSearchButton() {

ApplicationController.orderController.actionSearchButton(textField.getText().trim(), table);

refreshComboBoxes();

}

public void refreshView() {

ApplicationController.orderController.actionSearchButton("", table);

refreshComboBoxes();

}

public void refreshComboBoxes() {

try {

comboBox.setModel(

new DefaultComboBoxModel(ApplicationController.employeeController.getDAO().readAll().toArray()));

comboBox\_1.setModel(

new DefaultComboBoxModel(ApplicationController.clientController.getDAO().readAll().toArray()));

comboBox\_2.setModel(new DefaultComboBoxModel(

ApplicationController.warehouseController.getDAO().readAllWithCountMoreThenZero().toArray()));

} catch (Exception e) {

e.printStackTrace();

}

}

}

package forms;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JTable;

import controller.ApplicationController;

/\*\*

\* форма главного меню

\* @author admin

\*

\*/

public class MainFrame extends JFrame{

public MainFrame() {

initialize();

}

private void initialize() {

setTitle("Главная");

setBounds(100, 100, 233, 292);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getContentPane().setLayout(null);

JButton btnNewButton = new JButton("Заказы");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionOrderButton();

}

});

btnNewButton.setFont(new Font("Times New Roman", Font.PLAIN, 14));

btnNewButton.setBounds(10, 11, 195, 25);

getContentPane().add(btnNewButton);

JButton btnNewButton\_1 = new JButton("Товары");

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionProductButton();

}

});

btnNewButton\_1.setFont(new Font("Times New Roman", Font.PLAIN, 14));

btnNewButton\_1.setBounds(10, 115, 195, 25);

getContentPane().add(btnNewButton\_1);

JButton btnNewButton\_2 = new JButton("Поставщики");

btnNewButton\_2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionProviderButton();

}

});

btnNewButton\_2.setFont(new Font("Times New Roman", Font.PLAIN, 14));

btnNewButton\_2.setBounds(10, 79, 195, 25);

getContentPane().add(btnNewButton\_2);

JButton btnNewButton\_3 = new JButton("Клиенты");

btnNewButton\_3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionClientButton();

}

});

btnNewButton\_3.setFont(new Font("Times New Roman", Font.PLAIN, 14));

btnNewButton\_3.setBounds(10, 181, 195, 25);

getContentPane().add(btnNewButton\_3);

JButton btnNewButton\_4 = new JButton("Сотрудники");

btnNewButton\_4.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionEmployeeButton();

}

});

btnNewButton\_4.setFont(new Font("Times New Roman", Font.PLAIN, 14));

btnNewButton\_4.setBounds(10, 147, 195, 25);

getContentPane().add(btnNewButton\_4);

JButton btnNewButton\_5 = new JButton("Поставка");

btnNewButton\_5.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionDeliveryButton();

}

});

btnNewButton\_5.setFont(new Font("Times New Roman", Font.PLAIN, 14));

btnNewButton\_5.setBounds(10, 47, 195, 25);

getContentPane().add(btnNewButton\_5);

JButton btnNewButton\_6 = new JButton("Склад");

btnNewButton\_6.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionWarehouseButton();

}

});

btnNewButton\_6.setFont(new Font("Times New Roman", Font.PLAIN, 14));

btnNewButton\_6.setBounds(10, 217, 195, 25);

getContentPane().add(btnNewButton\_6);

}

private void actionWarehouseButton() {

ApplicationController.mainController.actionWarehouseButton();

}

private void actionDeliveryButton() {

ApplicationController.mainController.actionDeliveryButton();

}

// Buttons actions

private void actionOrderButton() {

ApplicationController.mainController.actionOrderButton();

}

private void actionProductButton() {

ApplicationController.mainController.actionProductButton();

}

private void actionProviderButton() {

ApplicationController.mainController.actionProviderButton();

}

private void actionClientButton() {

ApplicationController.mainController.actionClientButton();

}

private void actionEmployeeButton() {

ApplicationController.mainController.actionEmployeeButton();

}

}

package forms;

import java.awt.BorderLayout;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import controller.ApplicationController;

import javax.swing.JLabel;

import javax.swing.JTextField;

import javax.swing.JButton;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.JScrollPane;

import java.awt.Component;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import javax.swing.JTable;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

/\*\*

\* форма отображения сотрудников

\* @author admin

\*

\*/

public class EmployeeFrame extends JFrame {

private JTextField textField;

private JTextField textField\_1;

private JTextField textField\_2;

private JTextField textField\_3;

private JTable table;

public EmployeeFrame() {

setTitle("Сотрудник");

setBounds(100, 100, 592, 428);

getContentPane().setLayout(new BorderLayout(0, 0));

JPanel panel = new JPanel();

getContentPane().add(panel, BorderLayout.NORTH);

JLabel lblNewLabel\_4 = new JLabel("Введите ФИО для поиска");

panel.add(lblNewLabel\_4);

textField = new JTextField();

textField.setColumns(10);

panel.add(textField);

JButton btnNewButton\_1 = new JButton("Поиск");

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionSearchButton();

}

});

panel.add(btnNewButton\_1);

JPanel panel\_1 = new JPanel();

panel\_1.setBorder(new EmptyBorder(3, 3, 3, 3));

getContentPane().add(panel\_1, BorderLayout.WEST);

JLabel lblNewLabel = new JLabel("ФИО");

JLabel lblNewLabel\_1 = new JLabel("Телефон");

JLabel lblNewLabel\_2 = new JLabel("Email");

textField\_1 = new JTextField();

textField\_1.setColumns(10);

textField\_2 = new JTextField();

textField\_2.setColumns(10);

textField\_3 = new JTextField();

textField\_3.setColumns(10);

JButton btnNewButton\_2 = new JButton("Обновить");

btnNewButton\_2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionUpdateButton();

}

});

JButton btnNewButton = new JButton("Добавить");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionCreateButton();

}

});

JButton btnNewButton\_3 = new JButton("Удалить");

btnNewButton\_3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionDeleteButton();

}

});

GroupLayout gl\_panel\_1 = new GroupLayout(panel\_1);

gl\_panel\_1.setHorizontalGroup(

gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup()

.addContainerGap()

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.TRAILING)

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING, false)

.addComponent(btnNewButton\_3, GroupLayout.DEFAULT\_SIZE, 130, Short.MAX\_VALUE)

.addComponent(btnNewButton, GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(btnNewButton\_2, GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING, false)

.addGroup(gl\_panel\_1.createSequentialGroup()

.addGap(4)

.addComponent(lblNewLabel))

.addGroup(gl\_panel\_1.createSequentialGroup()

.addGap(4)

.addComponent(lblNewLabel\_1))

.addGroup(gl\_panel\_1.createSequentialGroup()

.addGap(4)

.addComponent(lblNewLabel\_2))

.addComponent(textField\_1, GroupLayout.DEFAULT\_SIZE, 130, Short.MAX\_VALUE)

.addComponent(textField\_2, GroupLayout.DEFAULT\_SIZE, 130, Short.MAX\_VALUE)

.addComponent(textField\_3, GroupLayout.DEFAULT\_SIZE, 130, Short.MAX\_VALUE)))

.addContainerGap(26, Short.MAX\_VALUE))

);

gl\_panel\_1.setVerticalGroup(

gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup()

.addComponent(lblNewLabel)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(textField\_1, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE)

.addGap(16)

.addComponent(lblNewLabel\_1)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(textField\_2, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE)

.addGap(16)

.addComponent(lblNewLabel\_2)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(textField\_3, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE)

.addGap(18)

.addComponent(btnNewButton)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(btnNewButton\_2)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(btnNewButton\_3)

.addGap(233))

);

panel\_1.setLayout(gl\_panel\_1);

table = new JTable();

table.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

actionTableMouseClicked();

}

});

getContentPane().add(new JScrollPane(table), BorderLayout.CENTER);

}

private void actionDeleteButton() {

if(table.getRowCount()>0 && table.getSelectedRowCount()>0) {

int row = table.getSelectedRow();

String id=table.getModel().getValueAt(row, 0).toString();

ApplicationController.employeeController.actionDeleteButton(Long.parseLong(id));

}

}

private void actionTableMouseClicked() {

if (table.getRowCount() > 0)

try {

int row = table.getSelectedRow();

textField\_1.setText(table.getModel().getValueAt(row, 1).toString());

textField\_2.setText(table.getModel().getValueAt(row, 2).toString());

textField\_3.setText(table.getModel().getValueAt(row, 3).toString());

} catch (Exception e) { }

}

private void actionUpdateButton() {

int column = 0;

int row = table.getSelectedRow();

Long id = Long.parseLong(table.getModel().getValueAt(row, column).toString());

ApplicationController.employeeController.actionUpdateButton(textField\_1.getText().trim(), textField\_2.getText().trim(), textField\_3.getText().trim(), id);

}

private void actionCreateButton() {

ApplicationController.employeeController.actionCreateButton(textField\_1.getText().trim(), textField\_2.getText().trim(), textField\_3.getText().trim());

}

private void actionSearchButton() {

ApplicationController.employeeController.actionSearchButton(textField.getText().trim(), table);

}

public void refreshView() {

ApplicationController.employeeController.actionSearchButton("", table);

}

}

package forms;

import java.awt.BorderLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.sql.Date;

import javax.swing.DefaultComboBoxModel;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.border.EmptyBorder;

import com.toedter.calendar.JDateChooser;

import controller.ApplicationController;

import entity.Delivery;

import entity.Product;

import entity.Provider;

import tableModal.DeliveryTableModal;

/\*\*

\* форма отображения поставок

\* @author admin

\*

\*/

public class DeliveryFrame extends JFrame {

private Long id\_product, id\_provider;

private JTextField textField;

private JTextField textField\_1;

private JTable table;

private JDateChooser dateChooser;

JComboBox<Product> comboBox;

JComboBox<Provider> comboBox\_1;

public DeliveryFrame() {

setTitle("Поставка");

setBounds(100, 100, 592, 428);

getContentPane().setLayout(new BorderLayout(0, 0));

JPanel panel = new JPanel();

getContentPane().add(panel, BorderLayout.NORTH);

JLabel lblNewLabel\_4 = new JLabel("Введите название продукта для поиска");

panel.add(lblNewLabel\_4);

textField = new JTextField();

textField.setColumns(10);

panel.add(textField);

JButton btnNewButton\_1 = new JButton("Поиск");

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionSearchButton();

}

});

panel.add(btnNewButton\_1);

JPanel panel\_1 = new JPanel();

panel\_1.setBorder(new EmptyBorder(3, 3, 3, 3));

getContentPane().add(panel\_1, BorderLayout.WEST);

JLabel lblNewLabel = new JLabel("Продукт");

JLabel lblNewLabel\_1 = new JLabel("Поставщик");

JLabel lblNewLabel\_2 = new JLabel("Дата");

textField\_1 = new JTextField();

textField\_1.setColumns(10);

textField\_1.addKeyListener(new KeyListener() {

@Override

public void keyTyped(KeyEvent keyEvent) {

char c = keyEvent.getKeyChar();

if (!(Character.isDigit(c)) || c == KeyEvent.VK\_BACK\_SPACE || c == KeyEvent.VK\_DELETE) {

keyEvent.consume();

}

}

@Override

public void keyReleased(KeyEvent keyEvent) {

}

@Override

public void keyPressed(KeyEvent e) {

// TODO Auto-generated method stub

}

});

JButton btnNewButton\_2 = new JButton("Обновить");

btnNewButton\_2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionUpdateButton();

}

});

JButton btnNewButton = new JButton("Добавить");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionCreateButton();

}

});

JButton btnNewButton\_3 = new JButton("Удалить");

btnNewButton\_3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionDeleteButton();

}

});

JLabel lblNewLabel\_2\_1 = new JLabel("Количество");

dateChooser = new JDateChooser();

dateChooser.setDateFormatString("yyyy-MM-dd");

comboBox = new JComboBox();

comboBox\_1 = new JComboBox();

GroupLayout gl\_panel\_1 = new GroupLayout(panel\_1);

gl\_panel\_1.setHorizontalGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING).addGroup(gl\_panel\_1

.createSequentialGroup().addContainerGap()

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING).addGroup(Alignment.TRAILING,

gl\_panel\_1.createSequentialGroup().addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(

gl\_panel\_1.createSequentialGroup().addGap(4).addComponent(lblNewLabel))

.addGroup(gl\_panel\_1.createSequentialGroup().addGap(4)

.addComponent(lblNewLabel\_1))

.addGroup(gl\_panel\_1.createSequentialGroup().addGap(4)

.addComponent(lblNewLabel\_2)))

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING, false)

.addComponent(btnNewButton\_3, GroupLayout.DEFAULT\_SIZE, 130, Short.MAX\_VALUE)

.addComponent(btnNewButton, GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE,

Short.MAX\_VALUE)

.addComponent(btnNewButton\_2, GroupLayout.DEFAULT\_SIZE,

GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addComponent(lblNewLabel\_2\_1)).addContainerGap(26, Short.MAX\_VALUE))

.addGroup(Alignment.TRAILING, gl\_panel\_1.createSequentialGroup()

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.TRAILING)

.addComponent(comboBox, Alignment.LEADING, 0, 130, Short.MAX\_VALUE)

.addComponent(dateChooser, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE, 130,

Short.MAX\_VALUE)

.addComponent(textField\_1, GroupLayout.DEFAULT\_SIZE, 130, Short.MAX\_VALUE))

.addGap(26))

.addGroup(gl\_panel\_1.createSequentialGroup()

.addComponent(comboBox\_1, GroupLayout.PREFERRED\_SIZE, 130, GroupLayout.PREFERRED\_SIZE)

.addContainerGap(26, Short.MAX\_VALUE)))));

gl\_panel\_1.setVerticalGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup().addComponent(lblNewLabel).addGap(2)

.addComponent(comboBox, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addGap(18).addComponent(lblNewLabel\_1).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(comboBox\_1, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addGap(14).addComponent(lblNewLabel\_2).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(dateChooser, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(ComponentPlacement.UNRELATED).addComponent(lblNewLabel\_2\_1)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(textField\_1, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addGap(24).addComponent(btnNewButton).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(btnNewButton\_2).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(btnNewButton\_3).addContainerGap(42, Short.MAX\_VALUE)));

panel\_1.setLayout(gl\_panel\_1);

table = new JTable();

table.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

actionTableMouseClicked();

}

});

getContentPane().add(new JScrollPane(table), BorderLayout.CENTER);

dateChooser.setDate(new java.util.Date());

}

private void actionDeleteButton() {

if (table.getRowCount() > 0 && table.getSelectedRowCount() > 0) {

int row = table.getSelectedRow();

String id = table.getModel().getValueAt(row, 0).toString();

ApplicationController.deliveryController.actionDeleteButton(Long.parseLong(id));

}

}

private void actionTableMouseClicked() {

if (table.getRowCount() > 0)

try {

int row = table.getSelectedRow();

Delivery delivery = (Delivery) table.getModel().getValueAt(row, DeliveryTableModal.ENTITY\_COL);

textField\_1.setText(delivery.getCount() + "");

Date date = Date.valueOf(delivery.getDate().toString());

dateChooser.setDate(date);

comboBox.getModel().setSelectedItem(delivery.getProduct());

comboBox\_1.getModel().setSelectedItem(delivery.getProvider());

} catch (Exception e) {

}

}

private void actionUpdateButton() {

int row = table.getSelectedRow();

Long id = Long.parseLong(table.getModel().getValueAt(row, DeliveryTableModal.ID\_COL).toString());

id\_product = ((Product) (comboBox.getSelectedItem())).getId();

id\_provider = ((Provider) (comboBox\_1.getSelectedItem())).getId();

ApplicationController.deliveryController.actionUpdateButton(id\_product, id\_provider,

new Date(dateChooser.getDate().getTime()), Integer.parseInt(textField\_1.getText()), id);

}

private void actionCreateButton() {

id\_product = Long.parseLong(comboBox.getSelectedItem().toString().split(":")[0]);

id\_provider = Long.parseLong(comboBox\_1.getSelectedItem().toString().split(":")[0]);

ApplicationController.deliveryController.actionCreateButton(id\_product, id\_provider,

new Date(dateChooser.getDate().getTime()), Integer.parseInt(textField\_1.getText()));

}

private void actionSearchButton() {

ApplicationController.deliveryController.actionSearchButton(textField.getText().trim(), table);

refreshComboBoxes();

}

public void refreshView() {

ApplicationController.deliveryController.actionSearchButton("", table);

refreshComboBoxes();

}

public void refreshComboBoxes() {

try {

comboBox.setModel(

new DefaultComboBoxModel(ApplicationController.productController.getDAO().readAll().toArray()));

comboBox\_1.setModel(

new DefaultComboBoxModel(ApplicationController.providerController.getDAO().readAll().toArray()));

} catch (Exception e) {

e.printStackTrace();

}

}

}

package forms;

import java.awt.BorderLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.sql.Date;

import javax.swing.DefaultComboBoxModel;

import javax.swing.GroupLayout;

import javax.swing.GroupLayout.Alignment;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.border.EmptyBorder;

import com.toedter.calendar.JDateChooser;

import controller.ApplicationController;

import entity.Delivery;

import entity.Product;

import entity.Provider;

import tableModal.DeliveryTableModal;

/\*\*

\* форма отображения поставок

\* @author admin

\*

\*/

public class DeliveryFrame extends JFrame {

private Long id\_product, id\_provider;

private JTextField textField;

private JTextField textField\_1;

private JTable table;

private JDateChooser dateChooser;

JComboBox<Product> comboBox;

JComboBox<Provider> comboBox\_1;

public DeliveryFrame() {

setTitle("Поставка");

setBounds(100, 100, 592, 428);

getContentPane().setLayout(new BorderLayout(0, 0));

JPanel panel = new JPanel();

getContentPane().add(panel, BorderLayout.NORTH);

JLabel lblNewLabel\_4 = new JLabel("Введите название продукта для поиска");

panel.add(lblNewLabel\_4);

textField = new JTextField();

textField.setColumns(10);

panel.add(textField);

JButton btnNewButton\_1 = new JButton("Поиск");

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionSearchButton();

}

});

panel.add(btnNewButton\_1);

JPanel panel\_1 = new JPanel();

panel\_1.setBorder(new EmptyBorder(3, 3, 3, 3));

getContentPane().add(panel\_1, BorderLayout.WEST);

JLabel lblNewLabel = new JLabel("Продукт");

JLabel lblNewLabel\_1 = new JLabel("Поставщик");

JLabel lblNewLabel\_2 = new JLabel("Дата");

textField\_1 = new JTextField();

textField\_1.setColumns(10);

textField\_1.addKeyListener(new KeyListener() {

@Override

public void keyTyped(KeyEvent keyEvent) {

char c = keyEvent.getKeyChar();

if (!(Character.isDigit(c)) || c == KeyEvent.VK\_BACK\_SPACE || c == KeyEvent.VK\_DELETE) {

keyEvent.consume();

}

}

@Override

public void keyReleased(KeyEvent keyEvent) {

}

@Override

public void keyPressed(KeyEvent e) {

// TODO Auto-generated method stub

}

});

JButton btnNewButton\_2 = new JButton("Обновить");

btnNewButton\_2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionUpdateButton();

}

});

JButton btnNewButton = new JButton("Добавить");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionCreateButton();

}

});

JButton btnNewButton\_3 = new JButton("Удалить");

btnNewButton\_3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionDeleteButton();

}

});

JLabel lblNewLabel\_2\_1 = new JLabel("Количество");

dateChooser = new JDateChooser();

dateChooser.setDateFormatString("yyyy-MM-dd");

comboBox = new JComboBox();

comboBox\_1 = new JComboBox();

GroupLayout gl\_panel\_1 = new GroupLayout(panel\_1);

gl\_panel\_1.setHorizontalGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING).addGroup(gl\_panel\_1

.createSequentialGroup().addContainerGap()

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING).addGroup(Alignment.TRAILING,

gl\_panel\_1.createSequentialGroup().addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(

gl\_panel\_1.createSequentialGroup().addGap(4).addComponent(lblNewLabel))

.addGroup(gl\_panel\_1.createSequentialGroup().addGap(4)

.addComponent(lblNewLabel\_1))

.addGroup(gl\_panel\_1.createSequentialGroup().addGap(4)

.addComponent(lblNewLabel\_2)))

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING, false)

.addComponent(btnNewButton\_3, GroupLayout.DEFAULT\_SIZE, 130, Short.MAX\_VALUE)

.addComponent(btnNewButton, GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE,

Short.MAX\_VALUE)

.addComponent(btnNewButton\_2, GroupLayout.DEFAULT\_SIZE,

GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addComponent(lblNewLabel\_2\_1)).addContainerGap(26, Short.MAX\_VALUE))

.addGroup(Alignment.TRAILING, gl\_panel\_1.createSequentialGroup()

.addGroup(gl\_panel\_1.createParallelGroup(Alignment.TRAILING)

.addComponent(comboBox, Alignment.LEADING, 0, 130, Short.MAX\_VALUE)

.addComponent(dateChooser, Alignment.LEADING, GroupLayout.DEFAULT\_SIZE, 130,

Short.MAX\_VALUE)

.addComponent(textField\_1, GroupLayout.DEFAULT\_SIZE, 130, Short.MAX\_VALUE))

.addGap(26))

.addGroup(gl\_panel\_1.createSequentialGroup()

.addComponent(comboBox\_1, GroupLayout.PREFERRED\_SIZE, 130, GroupLayout.PREFERRED\_SIZE)

.addContainerGap(26, Short.MAX\_VALUE)))));

gl\_panel\_1.setVerticalGroup(gl\_panel\_1.createParallelGroup(Alignment.LEADING)

.addGroup(gl\_panel\_1.createSequentialGroup().addComponent(lblNewLabel).addGap(2)

.addComponent(comboBox, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addGap(18).addComponent(lblNewLabel\_1).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(comboBox\_1, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addGap(14).addComponent(lblNewLabel\_2).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(dateChooser, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(ComponentPlacement.UNRELATED).addComponent(lblNewLabel\_2\_1)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(textField\_1, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE,

GroupLayout.PREFERRED\_SIZE)

.addGap(24).addComponent(btnNewButton).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(btnNewButton\_2).addPreferredGap(ComponentPlacement.RELATED)

.addComponent(btnNewButton\_3).addContainerGap(42, Short.MAX\_VALUE)));

panel\_1.setLayout(gl\_panel\_1);

table = new JTable();

table.addMouseListener(new MouseAdapter() {

@Override

public void mouseClicked(MouseEvent e) {

actionTableMouseClicked();

}

});

getContentPane().add(new JScrollPane(table), BorderLayout.CENTER);

dateChooser.setDate(new java.util.Date());

}

private void actionDeleteButton() {

if (table.getRowCount() > 0 && table.getSelectedRowCount() > 0) {

int row = table.getSelectedRow();

String id = table.getModel().getValueAt(row, 0).toString();

ApplicationController.deliveryController.actionDeleteButton(Long.parseLong(id));

}

}

private void actionTableMouseClicked() {

if (table.getRowCount() > 0)

try {

int row = table.getSelectedRow();

Delivery delivery = (Delivery) table.getModel().getValueAt(row, DeliveryTableModal.ENTITY\_COL);

textField\_1.setText(delivery.getCount() + "");

Date date = Date.valueOf(delivery.getDate().toString());

dateChooser.setDate(date);

comboBox.getModel().setSelectedItem(delivery.getProduct());

comboBox\_1.getModel().setSelectedItem(delivery.getProvider());

} catch (Exception e) {

}

}

private void actionUpdateButton() {

int row = table.getSelectedRow();

Long id = Long.parseLong(table.getModel().getValueAt(row, DeliveryTableModal.ID\_COL).toString());

id\_product = ((Product) (comboBox.getSelectedItem())).getId();

id\_provider = ((Provider) (comboBox\_1.getSelectedItem())).getId();

ApplicationController.deliveryController.actionUpdateButton(id\_product, id\_provider,

new Date(dateChooser.getDate().getTime()), Integer.parseInt(textField\_1.getText()), id);

}

private void actionCreateButton() {

id\_product = Long.parseLong(comboBox.getSelectedItem().toString().split(":")[0]);

id\_provider = Long.parseLong(comboBox\_1.getSelectedItem().toString().split(":")[0]);

ApplicationController.deliveryController.actionCreateButton(id\_product, id\_provider,

new Date(dateChooser.getDate().getTime()), Integer.parseInt(textField\_1.getText()));

}

private void actionSearchButton() {

ApplicationController.deliveryController.actionSearchButton(textField.getText().trim(), table);

refreshComboBoxes();

}

public void refreshView() {

ApplicationController.deliveryController.actionSearchButton("", table);

refreshComboBoxes();

}

public void refreshComboBoxes() {

try {

comboBox.setModel(

new DefaultComboBoxModel(ApplicationController.productController.getDAO().readAll().toArray()));

comboBox\_1.setModel(

new DefaultComboBoxModel(ApplicationController.providerController.getDAO().readAll().toArray()));

} catch (Exception e) {

e.printStackTrace();

}

}

}

package entity;

public class Warehouse extends AbstractEntity{

private Product product;

private int count;

public Warehouse() {

super();

}

public Warehouse(Product product, int count) {

super();

this.product = product;

this.count = count;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

public int getCount() {

return count;

}

public void setCount(int count) {

this.count = count;

}

@Override

public String toString() {

return product.getName()+":"+count;

}

}

package entity;

public class Provider extends AbstractEntity{

private String name;

private String adres;

private String phone;

private String email;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getAdres() {

return adres;

}

public void setAdres(String adres) {

this.adres = adres;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public Provider() {

super();

// TODO Auto-generated constructor stub

}

public Provider(String name, String adres, String phone, String email) {

super();

this.name = name;

this.adres = adres;

this.phone = phone;

this.email = email;

}

@Override

public String toString() {

return id+": "+ name;

}

}

package entity;

public class Product extends AbstractEntity{

private String name;

private int price;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getPrice() {

return price;

}

public void setPrice(int price) {

this.price = price;

}

public Product() {

super();

}

public Product(String name, int price) {

super();

this.name = name;

this.price = price;

}

@Override

public String toString() {

return id+": "+name;

}

}

package entity;

import java.sql.Date;

public class Order extends AbstractEntity{

private Employee employee;

private Client client;

private Product product;

private Date date;

private int count;

public Order() {

super();

}

public Order(Employee employee, Client client, Product product, Date date, int count) {

super();

this.employee = employee;

this.client = client;

this.product = product;

this.date = date;

this.count = count;

}

public Employee getEmployee() {

return employee;

}

public void setEmployee(Employee employee) {

this.employee = employee;

}

public Client getClient() {

return client;

}

public void setClient(Client client) {

this.client = client;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public int getCount() {

return count;

}

public void setCount(int count) {

this.count = count;

}

}

package entity;

public class Employee extends AbstractEntity{

private String fio;

private String phone;

private String email;

public Employee() {

super();

}

public Employee(String fio, String phone, String email) {

super();

this.fio = fio;

this.phone = phone;

this.email = email;

}

public String getFio() {

return fio;

}

public void setFio(String fio) {

this.fio = fio;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

@Override

public String toString() {

return fio;

}

}

package entity;

import java.sql.Date;

public class Delivery extends AbstractEntity{

private Provider provider;

private Product product;

private Date date;

private int count;

public Delivery(Provider provider, Product product, Date date, int count) {

super();

this.provider = provider;

this.product = product;

this.date = date;

this.count = count;

}

public Delivery() {

super();

}

public Provider getProvider() {

return provider;

}

public void setProvider(Provider provider) {

this.provider = provider;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public int getCount() {

return count;

}

public void setCount(int count) {

this.count = count;

}

@Override

public String toString() {

return "Delivery [provider=" + provider + ", product=" + product + ", date=" + date + ", count=" + count + "]";

}

}

package entity;

public class Client extends AbstractEntity{

private String fio;

private String phone;

private String email;

public Client() {

super();

}

public Client(String fio, String phone, String email) {

super();

this.fio = fio;

this.phone = phone;

this.email = email;

}

public String getFio() {

return fio;

}

public void setFio(String fio) {

this.fio = fio;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

@Override

public String toString() {

return fio;

}

}

package entity;

import java.io.Serializable;

public class AbstractEntity implements Serializable{

private static final long serialVersionUID = 1L;

Long id;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

}

мpackage defaultOperation;

import javax.swing.JFrame;

/\*\*

\* класс, содержащий стандартные операции для всех форм

\* @author admin

\*

\*/

public class StandartFrameOperation {

JFrame frame;

public StandartFrameOperation(JFrame frame) {

this.frame = frame;

}

public void showFrame() {

frame.setVisible(true);

}

public void hideFrame() {

frame.setVisible(false);

}

public boolean isVisible() {

return frame.isVisible();

}

public void switchVisible() {

frame.setVisible(!isVisible());

}

public JFrame getFrame() {

return frame;

}

public void setFrame(JFrame frame) {

this.frame = frame;

}

}

package dao;

import java.io.FileInputStream;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

import java.util.Properties;

import controller.ApplicationController;

import entity.Product;

import entity.Warehouse;

public class WarehouseDAO {

private Connection myConn;

public WarehouseDAO() throws Exception {

Properties props = new Properties();

props.load(new FileInputStream("db.properties"));

String user = props.getProperty("user");

String password = props.getProperty("password");

String dburl = props.getProperty("dburl");

myConn = DriverManager.getConnection(dburl, user, password);

System.out.println("DB warehouse connection success");

}

public List<Warehouse> readAll() throws Exception {

List<Warehouse> list = new ArrayList<Warehouse>();

Statement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.createStatement();

myRs = myStmt.executeQuery("SELECT \* FROM warehouse");

while (myRs.next()) {

Warehouse tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public List<Warehouse> readAllWithCountMoreThenZero() throws Exception {

List<Warehouse> list = new ArrayList<Warehouse>();

Statement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.createStatement();

myRs = myStmt.executeQuery("SELECT \* FROM warehouse WHERE count>0");

while (myRs.next()) {

Warehouse tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public List<Warehouse> readAllByIdProduct(Long id\_product) throws Exception {

List<Warehouse> list = new ArrayList<Warehouse>();

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.prepareStatement("SELECT \* FROM warehouse WHERE id\_product = ?");

myStmt.setLong(1, id\_product);

myRs = myStmt.executeQuery();

while (myRs.next()) {

Warehouse tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public List<Warehouse> search(String productName) throws Exception {

List<Warehouse> list = new ArrayList<Warehouse>();

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

productName= "%" + productName+ "%";

myStmt = myConn.prepareStatement("SELECT \* FROM warehouse WHERE id\_product in (SELECT id FROM product WHERE name LIKE ?)");

myStmt.setString(1, productName);

myRs = myStmt.executeQuery();

while (myRs.next()) {

Warehouse tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void create(Warehouse entity) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn

.prepareStatement("INSERT INTO warehouse" + " (id\_product, count)" + " values ( ?, ?)");

myStmt.setLong(1, entity.getProduct().getId());

myStmt.setInt(2, entity.getCount());

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

public List<Warehouse> read(Long id) throws Exception {

List<Warehouse> list = new ArrayList<Warehouse>();

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn

.prepareStatement("SELECT \* FROM warehouse WHERE id=?");

myStmt.setLong(1, id);

myRs=myStmt.executeQuery();

while (myRs.next()) {

Warehouse tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void update(Warehouse entity) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("UPDATE warehouse SET id\_product=?, count=? WHERE id=?");

myStmt.setLong(1, entity.getProduct().getId());

myStmt.setInt(2, entity.getCount());

myStmt.setLong(3, entity.getId());

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

public void Delete(Long id) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("DELETE FROM warehouse WHERE id=?");

myStmt.setLong(1, id);

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

private Warehouse convertRowToEntity(ResultSet myRs) throws SQLException {

Long id = myRs.getLong("id");

Long id\_product = myRs.getLong("id\_product");

int count = myRs.getInt("count");

Product product = null;

try {

product = ApplicationController.productController.getDAO().read(id\_product).get(0);

} catch (Exception e) {

e.printStackTrace();

}

Warehouse temp = new Warehouse(product, count);

temp.setId(id);

return temp;

}

private static void close(Connection myConn, Statement myStmt, ResultSet myRs) throws SQLException {

if (myRs != null) {

myRs.close();

}

if (myStmt != null) {

myStmt.close();

}

if (myConn != null) {

myConn.close();

}

}

private void close(Statement myStmt, ResultSet myRs) throws SQLException {

close(null, myStmt, myRs);

}

private void close(Statement myStmt) throws SQLException {

close(null, myStmt, null);

}

}

package dao;

import java.io.FileInputStream;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

import java.util.Properties;

import entity.Provider;

public class ProviderDAO {

private Connection myConn;

public ProviderDAO() throws Exception {

Properties props = new Properties();

props.load(new FileInputStream("db.properties"));

String user = props.getProperty("user");

String password = props.getProperty("password");

String dburl = props.getProperty("dburl");

myConn = DriverManager.getConnection(dburl, user, password);

System.out.println("DB provider connection success");

}

public List<Provider> readAll() throws Exception {

List<Provider> list = new ArrayList<Provider>();

Statement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.createStatement();

myRs = myStmt.executeQuery("SELECT \* FROM provider");

while (myRs.next()) {

Provider tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public List<Provider> search(String name) throws Exception {

List<Provider> list = new ArrayList<Provider>();

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

name = "%" + name + "%";

myStmt = myConn.prepareStatement("SELECT \* FROM provider WHERE name LIKE ?");

myStmt.setString(1, name);

myRs = myStmt.executeQuery();

while (myRs.next()) {

Provider tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void create(Provider provider) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn

.prepareStatement("insert into provider" + " (name, adres,phone,email)" + " values (?, ?, ?, ?)");

myStmt.setString(1, provider.getName());

myStmt.setString(2, provider.getAdres());

myStmt.setString(3, provider.getPhone());

myStmt.setString(4, provider.getEmail());

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

public List<Provider> read(Long id) throws Exception {

List<Provider> list = new ArrayList<Provider>();

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn

.prepareStatement("SELECT \* FROM provider WHERE id=?");

myStmt.setLong(1, id);

myRs=myStmt.executeQuery();

while (myRs.next()) {

Provider tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void update(Provider provider) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("UPDATE provider SET name=?, adres=?, phone=?, email=? WHERE id=?");

myStmt.setString(1, provider.getName());

myStmt.setString(2, provider.getAdres());

myStmt.setString(3, provider.getPhone());

myStmt.setString(4, provider.getEmail());

myStmt.setLong(5, provider.getId());

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

public void Delete(Long id) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("DELETE FROM provider WHERE id=?");

myStmt.setLong(1, id);

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

private Provider convertRowToEntity(ResultSet myRs) throws SQLException {

Long id = myRs.getLong("id");

String name = myRs.getString("name");

String adres = myRs.getString("adres");

String phone = myRs.getString("phone");

String email = myRs.getString("email");

Provider temp = new Provider(name, adres, phone, email);

temp.setId(id);

return temp;

}

private static void close(Connection myConn, Statement myStmt, ResultSet myRs) throws SQLException {

if (myRs != null) {

myRs.close();

}

if (myStmt != null) {

myStmt.close();

}

if (myConn != null) {

myConn.close();

}

}

private void close(Statement myStmt, ResultSet myRs) throws SQLException {

close(null, myStmt, myRs);

}

private void close(Statement myStmt) throws SQLException {

close(null, myStmt, null);

}

}

package dao;

import java.io.FileInputStream;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

import java.util.Properties;

import entity.Product;

public class ProductDAO {

private Connection myConn;

public ProductDAO() throws Exception {

Properties props = new Properties();

props.load(new FileInputStream("db.properties"));

String user = props.getProperty("user");

String password = props.getProperty("password");

String dburl = props.getProperty("dburl");

myConn = DriverManager.getConnection(dburl, user, password);

System.out.println("DB product connection success");

}

public List<Product > readAll() throws Exception {

List<Product > list = new ArrayList<Product >();

Statement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.createStatement();

myRs = myStmt.executeQuery("SELECT \* FROM product");

while (myRs.next()) {

Product tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public List<Product > search(String name) throws Exception {

List<Product > list = new ArrayList<Product >();

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

name= "%" + name+ "%";

myStmt = myConn.prepareStatement("SELECT \* FROM product WHERE name LIKE ?");

myStmt.setString(1, name);

myRs = myStmt.executeQuery();

while (myRs.next()) {

Product tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void create(Product entity) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn

.prepareStatement("insert into product" + " (name, price)" + " values (?, ?)");

myStmt.setString(1, entity.getName());

myStmt.setInt(2, entity.getPrice());

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

public List<Product > read(Long id) throws Exception {

List<Product > list = new ArrayList<Product >();

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn

.prepareStatement("SELECT \* FROM product WHERE id=?");

myStmt.setLong(1, id);

myRs=myStmt.executeQuery();

while (myRs.next()) {

Product tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void update(Product entity) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("UPDATE product SET name=?, price=? WHERE id=?");

myStmt.setString(1, entity.getName());

myStmt.setInt(2, entity.getPrice());

myStmt.setLong(3, entity.getId());

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

public void Delete(Long id) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("DELETE FROM product WHERE id=?");

myStmt.setLong(1, id);

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

private Product convertRowToEntity(ResultSet myRs) throws SQLException {

Long id = myRs.getLong("id");

String name= myRs.getString("name");

int price = myRs.getInt("price");

Product temp = new Product (name, price);

temp.setId(id);

return temp;

}

private static void close(Connection myConn, Statement myStmt, ResultSet myRs) throws SQLException {

if (myRs != null) {

myRs.close();

}

if (myStmt != null) {

myStmt.close();

}

if (myConn != null) {

myConn.close();

}

}

private void close(Statement myStmt, ResultSet myRs) throws SQLException {

close(null, myStmt, myRs);

}

private void close(Statement myStmt) throws SQLException {

close(null, myStmt, null);

}

}

package dao;

import java.io.FileInputStream;

import java.sql.Connection;

import java.sql.Date;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

import java.util.Properties;

import controller.ApplicationController;

import entity.Client;

import entity.Employee;

import entity.Order;

import entity.Product;

import entity.Warehouse;

public class OrderDAO {

private Connection myConn;

public OrderDAO() throws Exception {

Properties props = new Properties();

props.load(new FileInputStream("db.properties"));

String user = props.getProperty("user");

String password = props.getProperty("password");

String dburl = props.getProperty("dburl");

myConn = DriverManager.getConnection(dburl, user, password);

System.out.println("DB orders connection success");

}

public List<Order> readAll() throws Exception {

List<Order> list = new ArrayList<Order>();

Statement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.createStatement();

myRs = myStmt.executeQuery("SELECT \* FROM orders");

while (myRs.next()) {

Order tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public List<Order> search(String productName) throws Exception {

List<Order> list = new ArrayList<Order>();

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

productName = "%" + productName + "%";

myStmt = myConn.prepareStatement(

"SELECT \* FROM orders WHERE id\_product in (SELECT id FROM product where name LIKE ?)");

myStmt.setString(1, productName);

myRs = myStmt.executeQuery();

while (myRs.next()) {

Order tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void create(Order entity) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("insert into orders"

+ " (id\_employee, id\_client, id\_product, dates, count)" + " values (?, ?, ?, ?,?)");

myStmt.setLong(1, entity.getEmployee().getId());

myStmt.setLong(2, entity.getClient().getId());

myStmt.setLong(3, entity.getProduct().getId());

myStmt.setDate(4, entity.getDate());

myStmt.setInt(5, entity.getCount());

myStmt.executeUpdate();

// past

Warehouse warehouse = ApplicationController.warehouseController.getDAO()

.readAllByIdProduct(entity.getProduct().getId()).get(0);

if (warehouse.getCount() - entity.getCount() <= 0) {

ApplicationController.warehouseController.getDAO().Delete(warehouse.getId());

} else {

warehouse.setCount(warehouse.getCount() - entity.getCount());

ApplicationController.warehouseController.getDAO().update(warehouse);

}

} finally {

close(myStmt);

}

}

public List<Order> read(Long id) throws Exception {

List<Order> list = new ArrayList<Order>();

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.prepareStatement("SELECT \* FROM orders WHERE id=?");

myStmt.setLong(1, id);

myRs = myStmt.executeQuery();

while (myRs.next()) {

Order tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void update(Order entity) throws Exception {

PreparedStatement myStmt = null;

try {

Order past = read(entity.getId()).get(0);

Warehouse warehouse = ApplicationController.warehouseController.getDAO()

.readAllByIdProduct(past.getProduct().getId()).get(0);

int countDifference = past.getCount() - entity.getCount();

myStmt = myConn.prepareStatement(

"UPDATE orders SET id\_employee=?, id\_client=?, id\_product=?, dates=?, count=? WHERE id=?");

myStmt.setLong(1, entity.getEmployee().getId());

myStmt.setLong(2, entity.getClient().getId());

myStmt.setLong(3, entity.getProduct().getId());

myStmt.setDate(4, entity.getDate());

myStmt.setInt(5, entity.getCount());

myStmt.setLong(6, entity.getId());

myStmt.executeUpdate();

warehouse.setCount(warehouse.getCount() + past.getCount());

ApplicationController.warehouseController.getDAO().update(warehouse);

List<Warehouse> readAllByIdProduct = ApplicationController.warehouseController.getDAO()

.readAllByIdProduct(entity.getProduct().getId());

Warehouse warehouse2 = readAllByIdProduct.get(0);

warehouse2.setCount(warehouse2.getCount() - entity.getCount());

ApplicationController.warehouseController.getDAO().update(warehouse2);

} finally {

close(myStmt);

}

}

public void Delete(Long id) throws Exception {

PreparedStatement myStmt = null;

try {

Order delivery = read(id).get(0);

myStmt = myConn.prepareStatement("DELETE FROM orders WHERE id=?");

myStmt.setLong(1, id);

myStmt.executeUpdate();

List<Warehouse> warehouses = ApplicationController.warehouseController.getDAO()

.readAllByIdProduct(delivery.getProduct().getId());

if (warehouses.size() < 1) {

ApplicationController.warehouseController.getDAO()

.create(new Warehouse(delivery.getProduct(), delivery.getCount()));

} else {

Warehouse warehouse = warehouses.get(0);

warehouse.setCount(warehouse.getCount() + delivery.getCount());

ApplicationController.warehouseController.getDAO().update(warehouse);

}

} finally {

close(myStmt);

}

}

private Order convertRowToEntity(ResultSet myRs) throws SQLException {

Product product = null;

Employee employee = null;

Client client = null;

Order temp = null;

try {

Long id = myRs.getLong("id");

Long id\_employee = myRs.getLong("id\_employee");

Long id\_client = myRs.getLong("id\_client");

Long id\_product = myRs.getLong("id\_product");

Date dates = myRs.getDate("dates");

int count = myRs.getInt("count");

employee = ApplicationController.employeeController.getDAO().read(id\_employee).get(0);

client = ApplicationController.clientController.getDAO().read(id\_client).get(0);

product = ApplicationController.productController.getDAO().read(id\_product).get(0);

temp = new Order(employee, client, product, dates, count);

temp.setId(id);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return temp;

}

private static void close(Connection myConn, Statement myStmt, ResultSet myRs) throws SQLException {

if (myRs != null) {

myRs.close();

}

if (myStmt != null) {

myStmt.close();

}

if (myConn != null) {

myConn.close();

}

}

private void close(Statement myStmt, ResultSet myRs) throws SQLException {

close(null, myStmt, myRs);

}

private void close(Statement myStmt) throws SQLException {

close(null, myStmt, null);

}

}

package dao;

import java.io.FileInputStream;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

import java.util.Properties;

import entity.Employee;

public class EmployeeDAO {

private Connection myConn;

public EmployeeDAO() throws Exception {

Properties props = new Properties();

props.load(new FileInputStream("db.properties"));

String user = props.getProperty("user");

String password = props.getProperty("password");

String dburl = props.getProperty("dburl");

myConn = DriverManager.getConnection(dburl, user, password);

System.out.println("DB employee connection success");

}

public List<Employee> readAll() throws Exception {

List<Employee> list = new ArrayList<Employee>();

Statement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.createStatement();

myRs = myStmt.executeQuery("SELECT \* FROM employee");

while (myRs.next()) {

Employee tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public List<Employee> search(String fio) throws Exception {

List<Employee> list = new ArrayList<Employee>();

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

fio= "%" + fio+ "%";

myStmt = myConn.prepareStatement("SELECT \* FROM employee WHERE fio LIKE ?");

myStmt.setString(1, fio);

myRs = myStmt.executeQuery();

while (myRs.next()) {

Employee tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void create(Employee entity) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn

.prepareStatement("insert into employee" + " (fio,phone,email)" + " values ( ?, ?, ?)");

myStmt.setString(1, entity.getFio());

myStmt.setString(2, entity.getPhone());

myStmt.setString(3, entity.getEmail());

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

public List<Employee> read(Long id) throws Exception {

List<Employee> list = new ArrayList<Employee>();

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn

.prepareStatement("SELECT \* FROM employee WHERE id=?");

myStmt.setLong(1, id);

myRs=myStmt.executeQuery();

while (myRs.next()) {

Employee tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void update(Employee entity) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("UPDATE employee SET fio=?, phone=?, email=? WHERE id=?");

myStmt.setString(1, entity.getFio());

myStmt.setString(2, entity.getPhone());

myStmt.setString(3, entity.getEmail());

myStmt.setLong(4, entity.getId());

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

public void Delete(Long id) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("DELETE FROM employee WHERE id=?");

myStmt.setLong(1, id);

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

private Employee convertRowToEntity(ResultSet myRs) throws SQLException {

Long id = myRs.getLong("id");

String fio = myRs.getString("fio");

String phone = myRs.getString("phone");

String email = myRs.getString("email");

Employee temp = new Employee (fio, phone, email);

temp.setId(id);

return temp;

}

private static void close(Connection myConn, Statement myStmt, ResultSet myRs) throws SQLException {

if (myRs != null) {

myRs.close();

}

if (myStmt != null) {

myStmt.close();

}

if (myConn != null) {

myConn.close();

}

}

private void close(Statement myStmt, ResultSet myRs) throws SQLException {

close(null, myStmt, myRs);

}

private void close(Statement myStmt) throws SQLException {

close(null, myStmt, null);

}

}

package dao;

import java.io.FileInputStream;

import java.sql.Connection;

import java.sql.Date;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

import java.util.Properties;

import controller.ApplicationController;

import entity.Delivery;

import entity.Product;

import entity.Provider;

import entity.Warehouse;

public class DeliveryDAO {

private Connection myConn;

public DeliveryDAO() throws Exception {

Properties props = new Properties();

props.load(new FileInputStream("db.properties"));

String user = props.getProperty("user");

String password = props.getProperty("password");

String dburl = props.getProperty("dburl");

myConn = DriverManager.getConnection(dburl, user, password);

System.out.println("DB delivery connection success");

}

public List<Delivery> readAll() throws Exception {

List<Delivery> list = new ArrayList<Delivery>();

Statement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.createStatement();

myRs = myStmt.executeQuery("SELECT \* FROM delivery");

while (myRs.next()) {

Delivery tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public List<Delivery> search(String productName) throws Exception {

List<Delivery> list = new ArrayList<Delivery>();

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

productName = "%" + productName + "%";

myStmt = myConn.prepareStatement(

"SELECT \* FROM delivery WHERE id\_product in (SELECT id FROM product where name LIKE ?)");

myStmt.setString(1, productName);

myRs = myStmt.executeQuery();

while (myRs.next()) {

Delivery tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void create(Delivery entity) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement(

"insert into delivery" + " (id\_product, id\_provider, dates, count)" + " values (?, ?, ?, ?)");

myStmt.setLong(1, entity.getProduct().getId());

myStmt.setLong(2, entity.getProvider().getId());

myStmt.setDate(3, entity.getDate());

myStmt.setInt(4, entity.getCount());

myStmt.executeUpdate();

// past

List<Warehouse> readAllByIdProduct = ApplicationController.warehouseController.getDAO()

.readAllByIdProduct(entity.getProduct().getId());

if (readAllByIdProduct.size() > 0) {

Warehouse warehouse = readAllByIdProduct.get(0);

warehouse.setCount(warehouse.getCount() + entity.getCount());

ApplicationController.warehouseController.getDAO().update(warehouse);

} else {

ApplicationController.warehouseController.getDAO()

.create(new Warehouse(entity.getProduct(), entity.getCount()));

}

} finally {

close(myStmt);

}

}

public List<Delivery> read(Long id) throws Exception {

List<Delivery> list = new ArrayList<Delivery>();

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.prepareStatement("SELECT \* FROM delivery WHERE id=?");

myStmt.setLong(1, id);

myRs = myStmt.executeQuery();

while (myRs.next()) {

Delivery tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void update(Delivery entity) throws Exception {

PreparedStatement myStmt = null;

try {

Delivery past = read(entity.getId()).get(0);

Warehouse warehouse = ApplicationController.warehouseController.getDAO()

.readAllByIdProduct(past.getProduct().getId()).get(0);

int countDifference = past.getCount() - entity.getCount();

myStmt = myConn

.prepareStatement("UPDATE delivery SET id\_product=?, id\_provider=?, dates=?, count=? WHERE id=?");

myStmt.setLong(1, entity.getProduct().getId());

myStmt.setLong(2, entity.getProvider().getId());

myStmt.setDate(3, entity.getDate());

myStmt.setInt(4, entity.getCount());

myStmt.setLong(5, entity.getId());

myStmt.executeUpdate();

// past

warehouse.setCount(warehouse.getCount() - past.getCount());

ApplicationController.warehouseController.getDAO().update(warehouse);

// new

List<Warehouse> readAllByIdProduct = ApplicationController.warehouseController.getDAO()

.readAllByIdProduct(entity.getProduct().getId());

if (readAllByIdProduct.size() > 0) {

Warehouse warehouse2 = readAllByIdProduct.get(0);

warehouse2.setCount(warehouse2.getCount() + entity.getCount());

ApplicationController.warehouseController.getDAO().update(warehouse2);

} else {

ApplicationController.warehouseController.getDAO()

.create(new Warehouse(entity.getProduct(), entity.getCount()));

}

} finally {

close(myStmt);

}

}

public void Delete(Long id) throws Exception {

PreparedStatement myStmt = null;

try {

Delivery delivery = read(id).get(0);

myStmt = myConn.prepareStatement("DELETE FROM delivery WHERE id=?");

Warehouse warehouse = ApplicationController.warehouseController.getDAO()

.readAllByIdProduct(delivery.getProduct().getId()).get(0);

myStmt.setLong(1, id);

myStmt.executeUpdate();

if (warehouse.getCount() - delivery.getCount() == 0)

ApplicationController.warehouseController.getDAO().Delete(warehouse.getId());

else {

warehouse.setCount(warehouse.getCount() - delivery.getCount());

ApplicationController.warehouseController.getDAO().update(warehouse);

}

} finally {

close(myStmt);

}

}

private Delivery convertRowToEntity(ResultSet myRs) throws SQLException {

Product product;

Delivery temp = null;

try {

Long id = myRs.getLong("id");

Long id\_product = myRs.getLong("id\_product");

Long id\_provider = myRs.getLong("id\_provider");

Date dates = myRs.getDate("dates");

int count = myRs.getInt("count");

product = ApplicationController.productController.getDAO().read(id\_product).get(0);

Provider provider = ApplicationController.providerController.getDAO().read(id\_provider).get(0);

temp = new Delivery(provider, product, dates, count);

temp.setId(id);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return temp;

}

private static void close(Connection myConn, Statement myStmt, ResultSet myRs) throws SQLException {

if (myRs != null) {

myRs.close();

}

if (myStmt != null) {

myStmt.close();

}

if (myConn != null) {

myConn.close();

}

}

private void close(Statement myStmt, ResultSet myRs) throws SQLException {

close(null, myStmt, myRs);

}

private void close(Statement myStmt) throws SQLException {

close(null, myStmt, null);

}

}

package dao;

import java.io.FileInputStream;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

import java.util.Properties;

import entity.Client;

import entity.Provider;

public class ClientDAO {

private Connection myConn;

public ClientDAO() throws Exception {

Properties props = new Properties();

props.load(new FileInputStream("db.properties"));

String user = props.getProperty("user");

String password = props.getProperty("password");

String dburl = props.getProperty("dburl");

myConn = DriverManager.getConnection(dburl, user, password);

System.out.println("DB client connection success");

}

public List<Client> readAll() throws Exception {

List<Client> list = new ArrayList<Client>();

Statement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.createStatement();

myRs = myStmt.executeQuery("SELECT \* FROM client");

while (myRs.next()) {

Client tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public List<Client> search(String fio) throws Exception {

List<Client> list = new ArrayList<Client>();

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

fio= "%" + fio+ "%";

myStmt = myConn.prepareStatement("SELECT \* FROM client WHERE fio LIKE ?");

myStmt.setString(1, fio);

myRs = myStmt.executeQuery();

while (myRs.next()) {

Client tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void create(Client entity) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn

.prepareStatement("insert into client" + " (fio,phone,email)" + " values ( ?, ?, ?)");

myStmt.setString(1, entity.getFio());

myStmt.setString(2, entity.getPhone());

myStmt.setString(3, entity.getEmail());

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

public List<Client> read(Long id) throws Exception {

List<Client> list = new ArrayList<Client>();

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn

.prepareStatement("SELECT \* FROM client WHERE id=?");

myStmt.setLong(1, id);

myRs=myStmt.executeQuery();

while (myRs.next()) {

Client tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void update(Client entity) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("UPDATE client SET fio=?, phone=?, email=? WHERE id=?");

myStmt.setString(1, entity.getFio());

myStmt.setString(2, entity.getPhone());

myStmt.setString(3, entity.getEmail());

myStmt.setLong(4, entity.getId());

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

public void Delete(Long id) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("DELETE FROM client WHERE id=?");

myStmt.setLong(1, id);

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

private Client convertRowToEntity(ResultSet myRs) throws SQLException {

Long id = myRs.getLong("id");

String fio = myRs.getString("fio");

String phone = myRs.getString("phone");

String email = myRs.getString("email");

Client temp = new Client (fio, phone, email);

temp.setId(id);

return temp;

}

private static void close(Connection myConn, Statement myStmt, ResultSet myRs) throws SQLException {

if (myRs != null) {

myRs.close();

}

if (myStmt != null) {

myStmt.close();

}

if (myConn != null) {

myConn.close();

}

}

private void close(Statement myStmt, ResultSet myRs) throws SQLException {

close(null, myStmt, myRs);

}

private void close(Statement myStmt) throws SQLException {

close(null, myStmt, null);

}

}

package controller;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import dao.WarehouseDAO;

import defaultOperation.StandartFrameOperation;

import entity.Product;

import entity.Warehouse;

import forms.WarehouseFrame;

import tableModal.ClientTableModal;

import tableModal.WarehouseTableModal;

public class WarehouseController extends StandartFrameOperation{

WarehouseDAO DAO;

public WarehouseController(JFrame frame) {

super(frame);

try {

DAO = new WarehouseDAO();

} catch (Exception e) {e.printStackTrace();}

}

public void actionSearchButton(String productName, JTable table) {

try {

List<Warehouse> list = null;

if (productName!= null && productName.trim().length() > 0)

list = DAO.search(productName);

else

list = DAO.readAll();

WarehouseTableModal model = new WarehouseTableModal(list);

table.setModel(model);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

public void actionCreateButton(Product product, int count) {

if (product!=null && count >= 0) {

Warehouse entity = new Warehouse(product,count);

try {

DAO.create(entity);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно добавлено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

}

public void actionUpdateButton(Product product, int count, Long id) {

try {

Warehouse entity= new Warehouse(product,count);

entity.setId(id);

DAO.update(entity);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно обновлено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

public void actionDeleteButton(long id) {

if(id>0)

try {

DAO.Delete(id);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно удалено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка. Возможно элемент используется в другой таблице" , "Ошибка", JOptionPane.ERROR\_MESSAGE);

System.out.println(e);

}

}

public void refrechView() {

((WarehouseFrame)getFrame()).refreshView();

}

public WarehouseDAO getDAO() {return DAO;}

}

package controller;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import dao.ProductDAO;

import dao.ProviderDAO;

import defaultOperation.StandartFrameOperation;

import entity.Provider;

import forms.ProviderFrame;

import tableModal.ProviderTableModal;

public class ProviderController extends StandartFrameOperation {

ProviderDAO DAO;

public ProviderController(JFrame frame) {

super(frame);

try {

DAO = new ProviderDAO();

} catch (Exception e) {e.printStackTrace();}

}

public void actionSearchButton(String name, JTable table) {

try {

List<Provider> list = null;

if (name != null && name.trim().length() > 0)

list = DAO.search(name);

else

list = DAO.readAll();

ProviderTableModal model = new ProviderTableModal(list);

table.setModel(model);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

public void actionCreateButton(String name, String adres, String phone, String email) {

if (name.length() > 0 && adres.length() > 0 && phone.length() > 0 && email.length() > 0) {

Provider provider = new Provider(name, adres, phone, email);

try {

DAO.create(provider);

((ProviderFrame) getFrame()).refreshView();

JOptionPane.showMessageDialog(getFrame(), "Успешно добавлено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

}

public void actionUpdateButton(String name, String adres, String phone, String email, Long id) {

if (name.length() > 0 && adres.length() > 0 && phone.length() > 0 && email.length() > 0 && id>0) {

try {

Provider provider = new Provider(name, adres, phone, email);

provider.setId(id);

DAO.update(provider);

((ProviderFrame) getFrame()).refreshView();

JOptionPane.showMessageDialog(getFrame(), "Успешно обновлено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

}

public void actionDeleteButton(long id) {

if(id>0)

try {

DAO.Delete(id);

((ProviderFrame) getFrame()).refreshView();

JOptionPane.showMessageDialog(getFrame(), "Успешно удалено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка. Возможно элемент используется в другой таблице" , "Ошибка", JOptionPane.ERROR\_MESSAGE);

System.out.println(e);

}

}

public ProviderDAO getDAO() {

return DAO;

}

}

package controller;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import dao.ProductDAO;

import defaultOperation.StandartFrameOperation;

import entity.Product;

import forms.ProductFrame;

import tableModal.ProductTableModal;

public class ProductController extends StandartFrameOperation{

public ProductDAO DAO;

public ProductController(JFrame frame) {

super(frame);

try {

DAO = new ProductDAO();

} catch (Exception e) {e.printStackTrace();}

}

public void actionSearchButton(String name, JTable table) {

try {

List<Product > list = null;

if (name!= null && name.trim().length() > 0)

list = DAO.search(name);

else

list = DAO.readAll();

ProductTableModal model = new ProductTableModal(list);

table.setModel(model);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

public void actionCreateButton(String name, int price) {

if (name!=null && name.length() > 0 && price>0) {

Product entity = new Product (name,price);

try {

DAO.create(entity);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно добавлено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

}

public void actionUpdateButton(String name, int price, Long id) {

if (name!= null && name.length() > 0 && price> 0 && id>0) {

try {

Product entity= new Product(name, price);

entity.setId(id);

DAO.update(entity);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно обновлено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

}

public void actionDeleteButton(long id) {

if(id>0)

try {

DAO.Delete(id);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно удалено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка. Возможно элемент используется в другой таблице" , "Ошибка", JOptionPane.ERROR\_MESSAGE);

System.out.println(e);

}

}

public void refrechView() {

((ProductFrame) getFrame()).refreshView();

}

public ProductDAO getDAO() {

return DAO;

}

}

package controller;

import java.sql.Date;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import dao.OrderDAO;

import defaultOperation.StandartFrameOperation;

import entity.Client;

import entity.Employee;

import entity.Order;

import entity.Product;

import forms.OrderFrame;

import tableModal.OrderTableModal;

public class OrderController extends StandartFrameOperation {

OrderDAO DAO;

public OrderController(JFrame frame) {

super(frame);

try {

DAO = new OrderDAO();

} catch (Exception e) {

e.printStackTrace();

}

}

public void actionSearchButton(String productName, JTable table) {

try {

List<Order> list = null;

if (productName != null && productName.trim().length() > 0)

list = DAO.search(productName);

else

list = DAO.readAll();

OrderTableModal model = new OrderTableModal(list);

table.setModel(model);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

public void actionCreateButton(Long id\_employee, Long id\_client, Long id\_product, Date date, int count) {

if (id\_product > 0 && id\_employee > 0 && id\_client > 0 && count > 0) {

Product product = null;

Employee employee = null;

Client client = null;

try {

employee = ApplicationController.employeeController.getDAO().read(id\_employee).get(0);

client = ApplicationController.clientController.getDAO().read(id\_client).get(0);

product = ApplicationController.productController.getDAO().read(id\_product).get(0);

} catch (Exception e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

Order entity = new Order(employee, client, product, date, count);

try {

DAO.create(entity);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно добавлено", "Успех",

JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

e.printStackTrace();

}

}

}

public void actionUpdateButton(Long id\_employee, Long id\_client, Long id\_product, Date date, int count, Long id) {

if (id\_product > 0 && id\_employee > 0 && id\_client > 0 && count > 0 && id > 0) {

Product product = null;

Employee employee = null;

Client client = null;

try {

employee = ApplicationController.employeeController.getDAO().read(id\_employee).get(0);

client = ApplicationController.clientController.getDAO().read(id\_client).get(0);

product = ApplicationController.productController.getDAO().read(id\_product).get(0);

Order entity = new Order(employee, client, product, date, count);

entity.setId(id);

DAO.update(entity);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно обновлено", "Успех",

JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

e.printStackTrace();

}

}

}

public void actionDeleteButton(long id) {

if (id > 0)

try {

DAO.Delete(id);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно удалено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка. Возможно элемент используется в другой таблице",

"Ошибка", JOptionPane.ERROR\_MESSAGE);

System.out.println(e);

}

}

public void refrechView() {

((OrderFrame) getFrame()).refreshView();

}

public OrderDAO getDAO() {

return DAO;

}

}

package controller;

import javax.swing.JFrame;

import defaultOperation.StandartFrameOperation;

public class MainController extends StandartFrameOperation {

public MainController(JFrame frame) {

super(frame);

}

public void actionOrderButton() {

ApplicationController.orderController.switchVisible();

}

public void actionProductButton() {

ApplicationController.productController.switchVisible();

}

public void actionProviderButton() {

ApplicationController.providerController.switchVisible();

}

public void actionClientButton() {

ApplicationController.clientController.switchVisible();

}

public void actionEmployeeButton() {

ApplicationController.employeeController.switchVisible();

}

public void actionDeliveryButton() {

ApplicationController.deliveryController.switchVisible();

}

public void actionWarehouseButton() {

ApplicationController.warehouseController.switchVisible();

}

}

package controller;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import dao.ClientDAO;

import dao.EmployeeDAO;

import defaultOperation.StandartFrameOperation;

import entity.Employee;

import forms.EmployeeFrame;

import tableModal.EmployeeTableModal;

public class EmployeeController extends StandartFrameOperation {

EmployeeDAO DAO;

public EmployeeController(JFrame frame) {

super(frame);

try {

DAO = new EmployeeDAO();

} catch (Exception e) {

e.printStackTrace();

}

}

public void actionSearchButton(String fio, JTable table) {

try {

List<Employee> list = null;

if (fio != null && fio.trim().length() > 0)

list = DAO.search(fio);

else

list = DAO.readAll();

EmployeeTableModal model = new EmployeeTableModal(list);

table.setModel(model);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

public void actionCreateButton(String fio, String phone, String email) {

if (fio.length() > 0 && phone.length() > 0 && email.length() > 0) {

Employee entity = new Employee(fio, phone, email);

try {

DAO.create(entity);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно добавлено", "Успех",

JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

}

public void actionUpdateButton(String fio, String phone, String email, Long id) {

if (fio.length() > 0 && phone.length() > 0 && email.length() > 0 && id > 0) {

try {

Employee entity = new Employee(fio, phone, email);

entity.setId(id);

DAO.update(entity);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно обновлено", "Успех",

JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

}

public void actionDeleteButton(long id) {

if (id > 0)

try {

DAO.Delete(id);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно удалено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка. Возможно элемент используется в другой таблице",

"Ошибка", JOptionPane.ERROR\_MESSAGE);

System.out.println(e);

}

}

public void refrechView() {

((EmployeeFrame) getFrame()).refreshView();

}

public EmployeeDAO getDAO() {return DAO;}

}

package controller;

import java.sql.Date;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import dao.DeliveryDAO;

import defaultOperation.StandartFrameOperation;

import entity.Delivery;

import entity.Product;

import entity.Provider;

import forms.DeliveryFrame;

import tableModal.DeliveryTableModal;

public class DeliveryController extends StandartFrameOperation{

DeliveryDAO DAO;

public DeliveryController(JFrame frame) {

super(frame);

try {

DAO = new DeliveryDAO();

} catch (Exception e) {e.printStackTrace();}

}

public void actionSearchButton(String productName, JTable table) {

try {

List<Delivery > list = null;

if (productName!= null && productName.trim().length() > 0)

list = DAO.search(productName);

else

list = DAO.readAll();

DeliveryTableModal model = new DeliveryTableModal(list);

table.setModel(model);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

public void actionCreateButton(Long id\_product,Long id\_provider,Date date, int count) {

if (id\_product > 0 && id\_provider > 0 && count > 0) {

Product product = null;

Provider provider = null;

try {

product = ApplicationController.productController.getDAO().read(id\_product).get(0);

provider = ApplicationController.providerController.getDAO().read(id\_provider).get(0);

} catch (Exception e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

Delivery entity = new Delivery(provider, product, date, count);

try {

DAO.create(entity);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно добавлено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

}

public void actionUpdateButton(Long id\_product,Long id\_provider,Date date, int count, Long id) {

if (id\_product > 0 && id\_provider > 0 && count > 0 && id>0) {

try {

Product product = ApplicationController.productController.getDAO().read(id\_product).get(0);

Provider provider = ApplicationController.providerController.getDAO().read(id\_provider).get(0);

Delivery entity= new Delivery(provider, product, date, count);

entity.setId(id);

DAO.update(entity);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно обновлено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

}

public void actionDeleteButton(long id) {

if(id>0)

try {

DAO.Delete(id);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно удалено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка. Возможно элемент используется в другой таблице" , "Ошибка", JOptionPane.ERROR\_MESSAGE);

System.out.println(e);

}

}

public void refrechView() {

((DeliveryFrame) getFrame()).refreshView();

}

public DeliveryDAO getDAO() {return DAO;}

}

package controller;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import dao.ClientDAO;

import defaultOperation.StandartFrameOperation;

import entity.Client;

import forms.ClientFrame;

import tableModal.ClientTableModal;

public class ClientController extends StandartFrameOperation{

ClientDAO DAO;

public ClientController(JFrame frame) {

super(frame);

try {

DAO = new ClientDAO();

} catch (Exception e) {e.printStackTrace();}

}

public void actionSearchButton(String fio, JTable table) {

try {

List<Client> list = null;

if (fio!= null && fio.trim().length() > 0)

list = DAO.search(fio);

else

list = DAO.readAll();

ClientTableModal model = new ClientTableModal(list);

table.setModel(model);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

public void actionCreateButton(String fio, String phone, String email) {

if (fio.length() > 0 && phone.length() > 0 && email.length() > 0) {

Client entity = new Client(fio, phone, email);

try {

DAO.create(entity);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно добавлено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

}

public void actionUpdateButton(String fio, String phone, String email, Long id) {

if (fio.length() > 0 && phone.length() > 0 && email.length() > 0 && id>0) {

try {

Client entity= new Client(fio, phone, email);

entity.setId(id);

DAO.update(entity);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно обновлено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка: " + e, "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

}

public void actionDeleteButton(long id) {

if(id>0)

try {

DAO.Delete(id);

refrechView();

JOptionPane.showMessageDialog(getFrame(), "Успешно удалено", "Успех", JOptionPane.INFORMATION\_MESSAGE);

} catch (Exception e) {

JOptionPane.showMessageDialog(getFrame(), "Ошибка. Возможно элемент используется в другой таблице" , "Ошибка", JOptionPane.ERROR\_MESSAGE);

System.out.println(e);

}

}

public void refrechView() {

((ClientFrame) getFrame()).refreshView();

}

public ClientDAO getDAO() {return DAO;}

}

package controller;

import forms.ClientFrame;

import forms.DeliveryFrame;

import forms.EmployeeFrame;

import forms.MainFrame;

import forms.OrderFrame;

import forms.ProductFrame;

import forms.ProviderFrame;

import forms.WarehouseFrame;

public class ApplicationController {

public static MainController mainController = new MainController(new MainFrame());

public static ClientController clientController = new ClientController(new ClientFrame());

public static OrderController orderController = new OrderController(new OrderFrame());

public static EmployeeController employeeController = new EmployeeController(new EmployeeFrame());

public static ProductController productController = new ProductController(new ProductFrame());

public static ProviderController providerController = new ProviderController(new ProviderFrame());

public static DeliveryController deliveryController = new DeliveryController(new DeliveryFrame());

public static WarehouseController warehouseController = new WarehouseController(new WarehouseFrame());

public static void main(String[] args) throws Exception {

mainController.showFrame();

}

}

# ПРИЛОЖЕНИЕ Б. КОД СОЗДАНИЯ БАЗЫ ДАННЫХ

create database sklad;

use sklad;

create table provider(

id int(11) not null AUTO\_INCREMENT,

name varchar(255),

adres varchar(255),

phone varchar(255),

email varchar(255),

primary key(id)

);

INSERT INTO provider(id, name, adres,phone,email) VALUES

(1,'Поставщик 1','космонавтов 25','88005553535','one@yandex.ru'),

(2,'Поставщик 2','ломоносова 72','80295487998','two@mail.ru'),

(3,'Поставщик 3','брехатино 228','80442184914','three@google.com'),

(4,'Поставщик 4','липелова 89','80254897651','Four@yahoo.de'),

(5,'Поставщик 5','кузьминино 351','80335487618','F@list.ru');

create table product(

id int(11) not null AUTO\_INCREMENT,

name varchar(255),

price int ,

primary key(id)

);

INSERT INTO product(id, name,price) VALUES

(1,'Молоко',200),

(2,'Спирт',998),

(3,'ПК',2600),

(4,'Стол березовый',50),

(5,'Ручки синие',1);

create table delivery(

id int(11) not null AUTO\_INCREMENT,

primary key(id),

id\_product int not null,

id\_provider int not null,

dates date,

count int not null,

foreign key (id\_product) references product(id),

foreign key (id\_provider) references provider(id)

);

INSERT INTO delivery(id,id\_product,id\_provider,dates,count) VALUES

(1,1,5,'2020-10-01',20),

(2,2,4,'2020-10-02',40),

(3,3,3,'2020-10-03',8),

(4,4,2,'2020-10-06',16),

(5,5,1,'2020-10-06',56);

create table warehouse(

id int(11) not null AUTO\_INCREMENT,

id\_product int not null,

count int not null,

primary key (id),

foreign key (id\_product) references product(id)

);

INSERT INTO warehouse(id,id\_product,count) VALUES

(1,1,56),

(2,2,16),

(3,3,8),

(4,4,40),

(5,5,20);

create table employee(

id int(11) not null AUTO\_INCREMENT,

primary key(id),

fio varchar(255) not null,

phone varchar(255) not null,

email varchar(255) not null

);

INSERT INTO employee(id,fio,phone,email) VALUES

(1,'Иван Васильевич Грозный','88005553535','one@mail.ru'),

(2,'Корней Минатович Тутенко','808005553668','two@yandex.ru'),

(3,'Михаил петрович Гарусь','808005554777','tree@google.ru');

create table client(

id int(11) not null AUTO\_INCREMENT,

primary key(id),

fio varchar(255) not null,

phone varchar(255) not null,

email varchar(255) not null

);

INSERT INTO client(id,fio,phone,email) VALUES

(1,'Будько Анна Семенович','88005453535','one2@mail.ru'),

(2,'Кагура Андрей Боярский','808005562668','two3@yandex.ru'),

(3,'Вдовец Александр Васильевич','808005558277','tree1@google.ru');

create table orders(

id int(11) not null AUTO\_INCREMENT,

primary key(id),

id\_employee int not null,

id\_client int not null,

id\_product int not null,

count int not null,

dates date,

foreign key (id\_employee) references employee(id),

foreign key (id\_client) references client(id),

foreign key (id\_product) references product(id)

);

INSERT INTO orders(id,id\_employee,id\_client,id\_product,count,dates) VALUES

(1,1,3,1,2,'2020-10-07'),

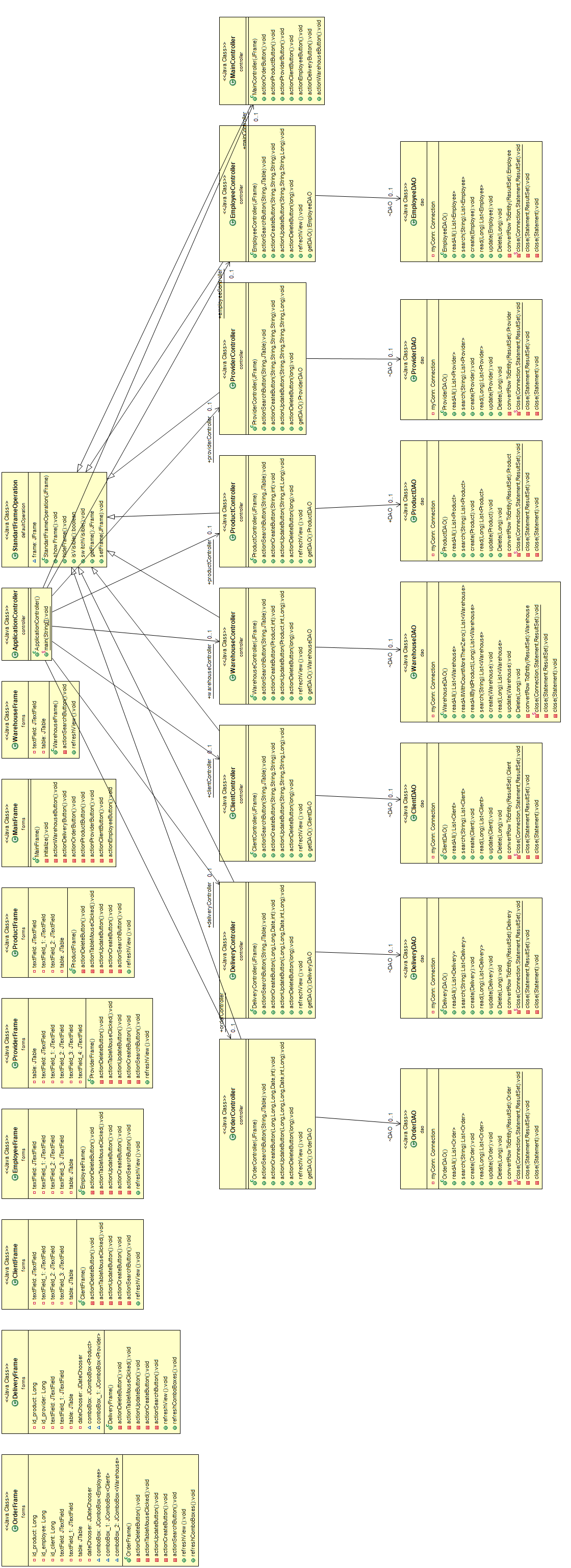
(2,2,2,2,2,'2020-10-07'),

(3,3,1,3,2,'2020-10-10'),

(4,1,3,4,3,'2020-10-11'),

(5,2,2,5,3,'2020-10-13');

# ПРИЛОЖЕНИЕ В. ДИАГРАММА КЛАССОВ. ЧАСТЬ 1



# ПРИЛОЖЕНИЕ Г. ДИАГРАММА КЛАССОВ. ЧАСТЬ 2

