# Содержание

[Введение 3](#_Toc56198721)

[1 Описание задачи на исследование и ее проектирование 4](#_Toc56198722)

[1.1 Формулировка и описание и задачи 4](#_Toc56198723)

[1.2 Разработка требования к программному продукту 4](#_Toc56198724)

[1.3 Проектирование программного продукта 7](#_Toc56198725)

[2. Реализация программного продукта 10](#_Toc56198726)

[2.1 Обоснование выбора инструментов для разработки 10](#_Toc56198727)

[2.2 Технология связи приложения с базой данных 12](#_Toc56198728)

[2.3 Анализ характерных частей кода 15](#_Toc56198729)

[3 Тестирование и развёртывание приложения 23](#_Toc56198730)

[3.1 Разработка плана тестирования 23](#_Toc56198731)

[3.2 Тестирование приложение 24](#_Toc56198732)

[3.3 Развертывание приложения 25](#_Toc56198733)

[Заключение 26](#_Toc56198734)

[Список использованных источников 27](#_Toc56198735)

[ПРИЛОЖЕНИЕ А – UML диаграмма 28](#_Toc56198736)

[ПРИЛОЖЕНИЕ Б – Схема базы данных 29](#_Toc56198737)

[ПРИЛОЖЕНИЕ В – Код программы 30](#_Toc56198738)

# Введение

В наше время, когда информационные технологии сопровождают человека на каждом шагу, сложно представить сферу деятельности, в которой не применена какая-либо автоматизация процессов. В данной курсовой работе речь пойдет об автоматизации процессов службы доставки лекарств. Важность такой службы очевидна. Лекарства необходимы в основном пожилым людям, которым, порой, сложно просто взять и пойти в аптеку, чтобы купить медикаменты, прописанные врачом. Однако, практически у каждого человека есть телефон для связи или компьютер с электронной почтой, с помощью которых, любой желающий может заказать доставку лекарственных средств на дом.

Целью курсовой работы является создание работоспособной системы для служб доставки лекарств, в которой будет просто вести учет заказов, следить за своевременной доставкой и наличием лекарств на складе. У программы должен быть интуитивно понятный интерфейс. Вся информация будет храниться в базе данных.

Основная задача состоит в том, чтобы создать программу, в которой будут храниться данные о покупателях, заказы покупателей на доставку, храниться информация о лекарствах и их доступном количестве.

Программа будет состоять из формы входа, основного окна для пользователя и администратора, а также вспомогательных окон, в которых, будут добавляться новые покупатели, добавляться/удаляться лекарства, доставка которых осуществляется на текущий момент и т.д.

# Описание задачи на исследование и ее проектирование

## Формулировка и описание и задачи

Сама экспресс-доставка появилась очень давно, конечно, она сильно отличается от современной доставки. Однако его суть была сохранена, включая доставку товара получателю в определенные сроки, сохранив при этом его первоначальный внешний вид. В современном мире весь процесс кажется гораздо более сложным, начиная с получения запроса на доставку и заканчивая состоянием "завершенной" доставки.

Учитывая конкретные обстоятельства деятельности организации, трудно вести учет без внедрения программного обеспечения. В бумажном виде такой учет займет много времени, чтобы заполнить, отследить и сохранить всю необходимую информацию. Невозможно или трудно выбрать из тех программ, которые уже имеются на рынке, которые будут отвечать всем требованиям и тонкостям.

Решение этой проблемы заключается в том, чтобы написать приложение для ваших собственных нужд и требований. Использование базы данных для хранения информации о деятельности вашей компании облегчит работу по сохранению накопленных данных путем создания резервных копий базы данных. Если компания переезжает в новый офис, то дешевле транспортировать компьютеры с базами данных и приложениями, нежели транспортировать картотечные шкафы и сейфы с бумажными носителями. Кроме того, анализ, обработка и использование такой базы данных менее ресурсоемки.

Исходя из этого, нужно написать приложение, которое позволит решить все бумажные вопросы, а также ускорить обработку данных и уменьшить количество ошибок. Программа должна представлять собой приложение, которое в зависимости от роли отображает пользователю свои интерактивные возможности.

## 1.2 Разработка требования к программному продукту

Доступ к программе должен осуществляться после ввода логина и пароля, которые будут выдаваться пользователям после запуска программы. На главной форме программы должно быть основное меню, которое позволит получить доступ к разрешенным для текущего пользователя разделам программы.

Главная форма должна иметь возможность зарегистрироваться или войти в свой аккаунт.

Форма пользователя должна содержать в себе возможность оформить новый заказ с выбором лекарств, их количества и временем доставки. Также нужно предусмотреть возможность пользователю просматривать текущие еще не доставленные заказы и уже доставленные.

Для работы администратора, на его форме должны быть расположены элементы взаимодействия со всеми таблицами. Нужно предусмотреть добавление/удаление элементов из таких таблиц как тип лекарств, лекарства, склад, пользователи, заказы.

Форма администратора также должна содержать информацию о лекарствах, у которых истекает срок годности.

Программа не должна хранить в своих модулях пользовательскую информацию. Вся информация введенная пользователем должна записываться в базу данных. Система управления базой данных должна быть MySQL.

Запуск программы сопровождается предложением ввести логин и пароль, или зарегистрироваться. Если пользователь выбирает регистрацию – он переходит на окно регистрации, где вводит свои данные, а именно фио, домашний адрес, номер телефона, после чего нажимает кнопку регистрация. Если регистрация произошла успешно, пользователь получает уведомление об этом и возвращается на страницу входа, если же нет – то получает соответствующее уведомление

После ввода данных необходимо нажать кнопку «Войти», данное событие сформирует запрос к базе данных для проверки введенных логина и пароля. Логин пользователя хранится в базе в открытом виде и сравнивается посимвольно.

Дальнейшая работа программы зависит от прав пользователя, под которым она была запущена. Если пользователь вводит неверные данные, программа предупреждает его об этом и просит повторно их ввести. После успешного ввода данных авторизации происходит проверка полномочий пользователя. Есть 2 вида пользователей: admin и user.

Если входит пользователь, открывается окно взаимодействия с пользователем, если администратор – окно администратора соответственно.

Форма работы с пользователем предоставляет ему на выбор меню с вариантами действий: «новый заказ», «текущие заказы.

При нажатии кнопки «новый заказ» будет осуществлен переход на форму оформление заказа. Она в себе содержит выбор из списка лекарства, количество этого лекарства, а также дату заказа.

На форме администратора можно перейти ко следующим формам работы с данными: типы лекарств, лекарства, лекарства на складе, пользователи, заказы, а также форма с информацией об лекарствах с истекающим сроком годности

Таким образом, мы имеем следующую диаграмму взаимодействия.

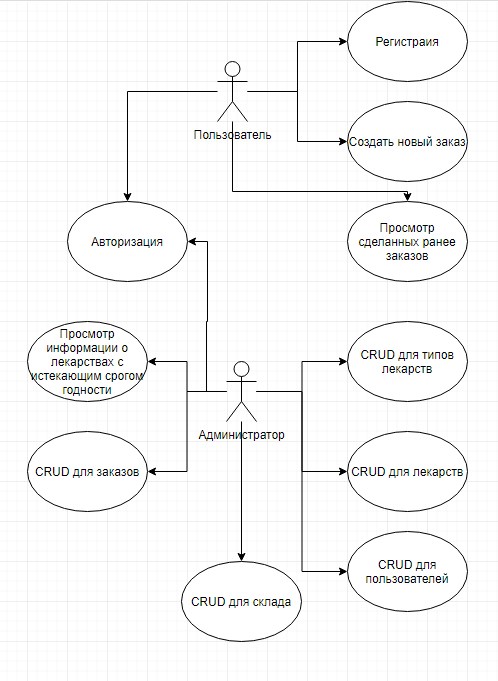


Рисунок 1.1 – UML диаграмма

## 1.3 Проектирование программного продукта

Для взаимодействия внутри приложения с базой данных, нам необходимо использовать классы-сущности для каждой из таблиц. Поскольку такие классы всецело повторяют типы полей из соответствующих таблиц, можем описать таблицу базы данных и сущность класса одного типа в одной таблице.

Так, у нас будут использоваться следующие таблицы и классы сущности:

Таблица 1 – MedicType (тип лекарства)

|  |  |
| --- | --- |
| **Название поля** | **Тип поля** |
| type | string |

Таблица, представляющая собой список типов для всех лекарст

Таблица 2 – Medic (лекарство)

|  |  |
| --- | --- |
| **Название поля** | **Тип поля** |
| Id | string |
| Name | String |
| type | String |
| sizeType | string |

Таблица, представляющая собой все лекарства, последний параметр – это объем одного лекарства, к примеру упаковка 100шт

Таблица 3 – Role (роли)

|  |  |
| --- | --- |
| **Название поля** | **Тип поля** |
| role | string |

Таблица ролей для пользователя, чтобы открывать форму в зависимости от роли.

Таблица 4 – Users (пользователь)

|  |  |
| --- | --- |
| **Название поля** | **Тип поля** |
| Id | long |
| Fio | String |
| Adres | String |
| Phone | String |
| role | string |

Таблица, представляющая собой список пользователей с их параметрами. В качестве типа данных для телефона используется строка, поскольку некоторые организации используют в поле номера телефона такие знаки как скобки, знака плюса.

Таблица 5 – Warehouse (склад)

|  |  |
| --- | --- |
| **Название поля** | **Тип поля** |
| id | Long |
| Medic | Int |
| Count | Int |
| dateGetting | Date |
| dateDead | Date |
| priceGetting | Float |
| priceSelling | Float |

Таблица, представляющая собой склад лекарств. Она дает данные о лекарстве на складе, его количестве, даты привоза на склад и даты истечения срока годности. Также дает информацию о цене закупке и цене продажи данного лекарства.

# 2. Реализация программного продукта

## 2.1 Обоснование выбора инструментов для разработки

Java-это объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems с 1991 года и официально выпущенный 23 мая 1995 года. 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

Отличительной особенностью Java по сравнению с другими универсальными языками программирования является то, что она обеспечивает высокую производительность программирования, а не эффективность работы приложений или использования памяти.

Java использует почти те же соглашения для объявления переменных, передачи параметров, операторов и управления потоком выполнения кода. Java добавляет Все хорошие возможности C++.

Технология языка Java сочетает в себе три ключевых элемента

Java предоставляет свои собственные апплеты для широкого использования-небольшие,надежные, динамичные, независимые от платформы, активные веб-приложения, встроенные в веб-страницы. Java-апплеты можно легко настроить и распространить среди потребителей, как и любой HTML-документ

Java высвобождает возможности объектно-ориентированной разработки приложений, сочетая простой и привычный синтаксис с мощной и простой в использовании средой разработки. Это позволяет широкому кругу программистов быстро создавать новые программы и новые апплеты

Java предоставляет программистам богатый набор классов объектов для четкого абстрагирования многих системных функций для Windows, сетей и ввода-вывода.Ключевой особенностью этих классов является то, что они предоставляют независимые от платформы абстракции для различных системных интерфейсов

Самое большое преимущество Java заключается в том, что вы можете использовать его для создания приложений, которые могут работать на разных платформах. К интернету подключены различные типы компьютеров-Pentium PC,Macintosh,Sun workstation и др. Даже в компьютерах на базе процессоров Intel существует несколько платформ, таких как Microsoft Windows версии 3.1, Windows95, Windows NT, OS/2, Solaris, различные операционные системы UNIX с графическими оболочками XWindows. В то же время, создавая веб-сервер в интернете, мы хотим, чтобы им пользовалось как можно больше людей. В этом случае помогут Java-приложения, предназначенные для работы на разных платформах и не зависящие от конкретного типа процессора и операционной системы.

Программы, написанные на языке программирования Java, можно разделить на две категории в зависимости от их использования.

К первой группе относятся Java-приложения, предназначенные для автономной работы под управлением специальных машин интерпретации Java. Реализация этой машины создана для всех основных машин. Вторая группа-это так называемые апплеты. Небольшая программа-это Java-приложение, которое интерпретируется виртуальной машиной Java, встроенной почти во все современные браузеры.

Приложения, относящиеся к первой группе, являются обычными автономными программами. Поскольку они не содержат машинного кода и работают под управлением специальных интерпретаторов, их производительность значительно ниже, чем у скомпилированных обычных программ, например, на языке программирования C++. Java-программы, не требующие повторной передачи, могут работать на любой платформе, что само по себе очень важно при разработке Интернета.

Поскольку AWT имеет недостатки, которые мы обсуждали ранее, Sun последовала за Awt, чтобы разработать набор графических компонентов, называемых Swing. Стоит отметить, что компоненты Swing полностью написаны на Java. 2D используется для рендеринга,и это время приносит несколько преимуществ. В результате набор стандартных компонентов значительно превосходит AWT по разнообразию и функциональности.

Элементы пользовательского интерфейса, такие как меню, кнопки и т. д., отображаются в пустом окне. Специфичные для платформы оконные системы должны только отображать окно и рисовать в нем графику.

Поэтому элементы графического интерфейса пользователя, созданные с помощью swing, выглядят и ведут себя точно так же, но не зависят от платформы, на которой запущена программа. Swing теперь является частью библиотеки Java base class (JFC).

Более тогоSun Microsystemsразработала независимый от платформы стиль по названием Metal, который сообщество программирующихна Javaпрозвало стилемJava.

Преимущества библиотеки Swing:

* содержит более богатый и удобный набор элементов пользовательского интерфейса
* намного меньше зависит от той платформы, на которой должна выполняться программа. Следовательно, меньше подвержена ошибкам, характерным для конкретной платформы
* обеспечивает одинаковое восприятие конечными пользователями приложений с ГПИ на разных платформах
* встроенный редактор форм почти во всех средах разработки
* на базе свинга есть много расширений типа SwingX
* поддержка различных стилей (Look and feel)

Недостатки:

* окно с множеством компонентов начинает подтормаживать;
* работа с менеджерами компоновки может стать настоящим кошмаром в сложных интерфейсах

Так как библиотека swing имеет больше возможностей для создания ГПИ, в дальнейшем мы будем использовать именно ее

## 2.2 Технология связи приложения с базой данных

JDBC (Java Database Connectivity) - это интерфейс, а не протокол, основанный на спецификации SAG CLI (SQL Access Group Call Level Interface).

Сам JDBC не работает и использует базовые абстракции и методы ODBC. Хотя стандарт API JDBC обеспечивает не только возможность работы через ODBC, но и возможность использования прямых ссылок на базы данных с использованием схем двойного или тройного связывания (см. Рис. 1), он сравнивается с широко используемым JDBC-ODBC-Bridge.，(см. Рис. 2.2)

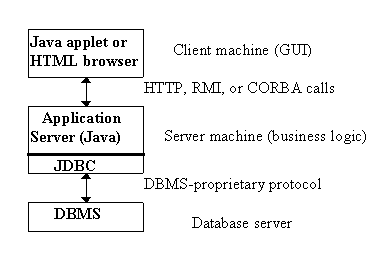


Рисунок 2.1 - Непосредственный доступ к базе данных по 3-х-звенной схеме.

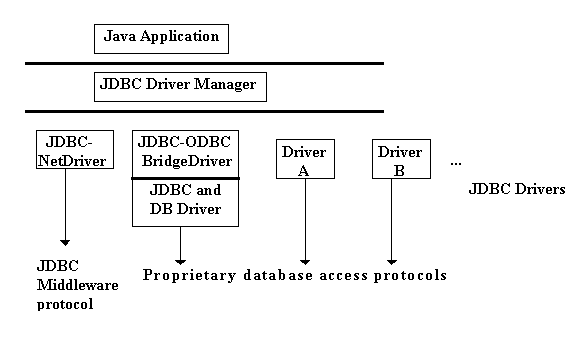


Рисунок 2.2 - Схема взаимодействия интерфейсов.

Даже беглого взгляда на Рис. 2 вполне достаточно, чтобы понять - общая схема взаимодействия интерфейсов в Java удивительным образом напоминает столь всем знакомую схему ODBC с ее гениальным изобретением драйвер-менеджера к различным СУБД и единого универсального пользовательского интерфейса. JDBC Driver Manager - это основной ствол JDBC-архитектуры. Его первичные функции очень просты - соединить Java-программу и соответствующий JDBC драйвер и затем выйти из игры. Естественно, что ODBC был взят в качестве основы JDBC из-за его популярности среди независимых поставщиков программного обеспечения и пользователей. Но тогда возникает законный вопрос - а зачем вообще нужен JDBC и не легче ли было организовать интерфейсный доступ к ODBC-драйверам непосредственно из Java? Ответом на этот вопрос может быть только однозначное нет. Путь через JDBC-ODBC-Bridge, как ни странно, может оказаться гораздо короче.

ODBC нельзя использовать непосредственно из Java, поскольку он основан на интерфейсе C. Вызов из кода Java C нарушает всю концепцию Java,нарушает уязвимости безопасности и затрудняет перенос программ.

Перенос ODBC C API на Java API не рекомендуется. Например, Java не имеет указателей, и ODBC использует их.

ODBC слишком сложен для понимания. Он смешивает простые и сложные вещи и иногда применяет сложные параметры к самым простым запросам.

Java API необходим для реализации абсолютно чистого Java-решения. При использовании ODBC необходимо установить драйвер ODBC и диспетчер ODBC на каждом клиентском компьютере. В то же время драйверы JDBC полностью написаны на Java и могут быть легко перенесены на любую платформу с сетевого компьютера на мэйнфрейм.

JDBC API - это естественный Java-интерфейс с базовой абстракцией SQL и принимает дух и базовую абстракцию концепции ODBC, в конце концов, он реализован как реальный Java-интерфейс, совместимый с остальной частью Java-системы.

В отличие от интерфейса ODBC, JDBC гораздо проще организовать. Основная его часть-это драйвер, предоставляемый JavaSoft для доступа к источникам данных из JDBC. Этот драйвер является самым высоким драйвером в иерархии классов JDBC и называется DriverManager. Согласно установленным правилам интернета, база данных и средства ее обслуживания идентифицируются с помощью URL-адресов.

jdbc

где под понимается имя конкретного драйвера, или некоего механизма установления соединения с базой данных, например, ODBC. В случае применения ODBC, в URL-строку подставляется именно эта аббревиатура, а в качестве используется обычный DSN (Data Source Name), т.е. имя ODBC-источника из ODBC.INI файла. Например:

jdbc:odbc:dBase

В некоторых случаях вместо ODBC может быть использовано имя прямого сетевого сервиса к базе данных, например:

jdbc:dcenaming:accounts-payable,

или

jdbc:dbnet://ultra1:1789/state

В последнем случае часть URL //ultra1:1789/state представляет собой и описывает имя хоста, порт и соответствующий идентификатор для доступа к соответствующей базе данных.

Однако, как уже говорилось выше, чаще всего, все-таки используется механизм ODBC благодаря его универсальности и доступности. Программа взаимодействия между драйвером JDBC и ODBC разработана фирмой JavaSoft в сотрудничестве с InterSolv и называется JDBC-ODBC-Bridge. Она реализована в виде JdbcOdbc.class (для платформы Windows JdbcOdbc.dll) и входит в поставку JDK1.1. Помимо JdbcOdbc-библиотек должны существовать специальные драйвера (библиотеки), которые реализуют непосредственный доступ к базам данных через стандартный интерфейс ODBC. Как правило эти библиотеки описываются в файле ODBC.INI. На внутреннем уровне JDBC-ODBC-Bridge отображает медоды Java в вызовы ODBC и тем самым позволяет использовать любые существующие драйверы ODBC, которых к настоящему времени накоплено в изобилии.

## 2.3 Анализ характерных частей кода

Все классы приложения делятся на части, в зависимости от функционала. Так, у нас есть несколько групп классов, которые работают подобным друг-другу образом. Группы:

* Контроллеры
* ДАО
* Сущности
* Формы отображения
* Модели отображения в таблице

Формы отвечают за отображение данных пользователю. К примеру, форма MedicTypeFrame отображает пользователю таблицу типов товаров с кнопками взаимодействия.

Для каждой кнопки формы предусмотрены свои методы

Листинг 1 – Код кнопок формы типов лекарств

//отображение данные в полях при нажатии на строку таблицы

private void actionTableMouseClicked() {

if (table.getRowCount() > 0)

try {

int row = table.getSelectedRow();

textField.setText(table.getModel().getValueAt(row, 0).toString());

} catch (Exception e) {

}

}

//кнопка удаления

private void actionDeleteButton() {

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

int row = table.getSelectedRow();

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

ApplicationController.MedicTypeController.actionDeleteButton(type);

}

}

//кнопка обновления

private void actionUpdateButton() {

int column = 0;

int row = table.getSelectedRow();

ApplicationController.MedicTypeController.actionUpdateButton(textField.getText(),table.getModel().getValueAt(row, 0).toString());

refreshView();

}

//кнопка создания

private void actionCreateButton() {

ApplicationController.MedicTypeController.actionCreateButton(textField.getText());

}

//кнопка поиска

private void actionSearchButton() {

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

}

Как видно из кода, для связи формы и функционала используются контроллеры, методы которого вызываются из формы

Листинг 2 – Контроллер типов лекарств

// конструктор

public MedicTypeController(JFrame frame) {

super(frame);

try {

DAO = new MedicTypeDAO();

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

}

// поиск по полям

public void actionSearchButton(String name, JTable table) {

try {

List<MedicType> list = null;

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

list = DAO.search(name);

else

list = DAO.readAll();

MedicTypeTableModal model = new MedicTypeTableModal(list);

table.setModel(model);

} catch (Exception e) {

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

}

}

// добавление

public void actionCreateButton(String type) {

if (type.length() > 0 ) {

MedicType mt = new MedicType(type);

try {

DAO.create(mt);

((MedicTypeFrame) getFrame()).refreshView();

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

} catch (Exception e) {

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

}

}

}

// обновление

public void actionUpdateButton(String typeNew, String typeOld) {

if (typeNew.length()>0 ) {

try {

MedicType mt = new MedicType(typeNew);

DAO.update(typeNew,typeOld);

((MedicTypeFrame) getFrame()).refreshView();

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

} catch (Exception e) {

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

}

}

}

// удаление

public void actionDeleteButton(String type) {

try {

DAO.Delete(type);

((MedicTypeFrame) 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);

}

}

Для работы с данными используются классы DAO, которые напрямую работают с бд. В них передаются сущности, с которыми идет работа

Листинг 3 – DAO типа лекарств

// конструктор, подключающийся к бд

public MedicTypeDAO() 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 medicType connection success");

}

// чтение всех записей

public List<MedicType> readAll() throws Exception {

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

Statement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.createStatement();

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

while (myRs.next()) {

MedicType tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

// поиск элементов

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

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

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

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

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

myStmt.setString(1, name);

myRs = myStmt.executeQuery();

while (myRs.next()) {

MedicType tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

// создание записи

public void create(MedicType entity) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn

.prepareStatement("insert into type\_medic" + " (type)" + " values (?)");

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

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

// чтение определенной записи

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

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

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn

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

myStmt.setLong(1, id);

myRs=myStmt.executeQuery();

while (myRs.next()) {

MedicType tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

// обновление записи

public void update(String typeNew, String typeOld) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("UPDATE type\_medic SET type=? WHERE type=?");

myStmt.setString(1, typeNew);

myStmt.setString(2, typeOld);

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

// удаление записи

public void Delete(String type) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("DELETE FROM type\_medic WHERE type=?");

myStmt.setString(1, type);

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

// преоббразование сущности в строку таблицы для отображения

private MedicType convertRowToEntity(ResultSet myRs) throws SQLException {

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

MedicType temp = new MedicType(type);

return temp;

}

// закрытие всех соединений

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();

}

}

// закрытие соединений

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

close(null, myStmt, myRs);

}

// закрытие соединений

private void close(Statement myStmt) throws SQLException {

close(null, myStmt, null);

Модели таблиц используются только для правильного отображение столбцов таблицы.

# 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. 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 6 приведены результаты тестирования программного обеспечения. Используйте метод анализа граничных условий для тестирования.

Таблица 6– Test Case

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Описание Test Case | Ожидаемый  результат | Полученный  Результат |
| 1 | 1.Открыть программу  2. Войти как админ | Переход на страницу админа | Открывается форма админа |
| 1 | 1.Открыть программу  2. Войти как пользователь | Переход на страницу пользователя | Открывается форма пользователя |
| 2 | 1.Открыть программу  2.Ввести несуществующие данные | Невозможность входа | Появление сообщения об ошибке, невозможно куда-то войти |
| 3 | 1.Открыть программу  2. Войти как админ.  3.Перейти на страницу типа лекарства  4. Добавить тип | В таблицу добавился новый тип. В таблице | В таблице отображен новый тип. |
| 5 | 1.Открыть программу  2. Перейти к форме регистрации  3.Зарегистрироваться | Добавление нового пользователя | Пользователю показалось уведомление об успешной регистрации, сейчас можно войти в систему |

## 3.3 Развертывание приложения

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

* Виртуальная машина Java и Jdk 1.8;
* Среда разработки Eclipse;
* MySSL Server;

После того как пользователь установил данное программное обеспечение следует импортировать создать базу данных, использовав скрипт «SQL» 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 моделирование службы доставки лекарств.

При выполнении работы, были детально изучена используемая функциональность и возможности объектно-ориентированного языка Java.

Кроме того, работа способствовала приобредению навыков работы с Java Spring, JDBC, а также основные 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.

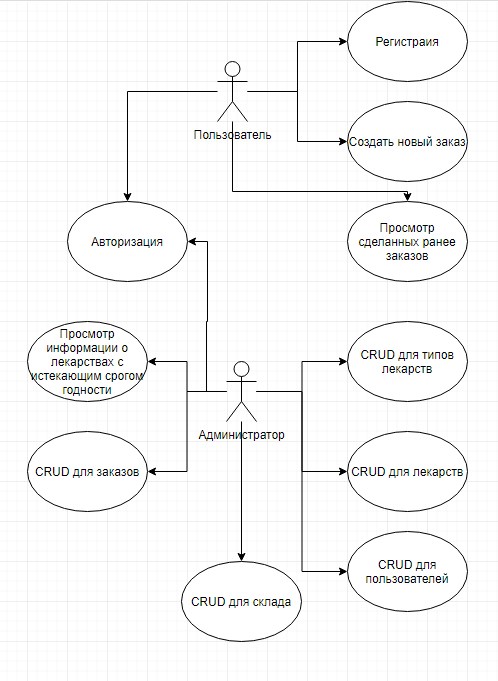
Программа соответствует исходному техническому заданию. В теории, ее можно использовать для большинства работ, связанных с медициной, в частности в организациях, занимающихся доставкой лекарств.

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

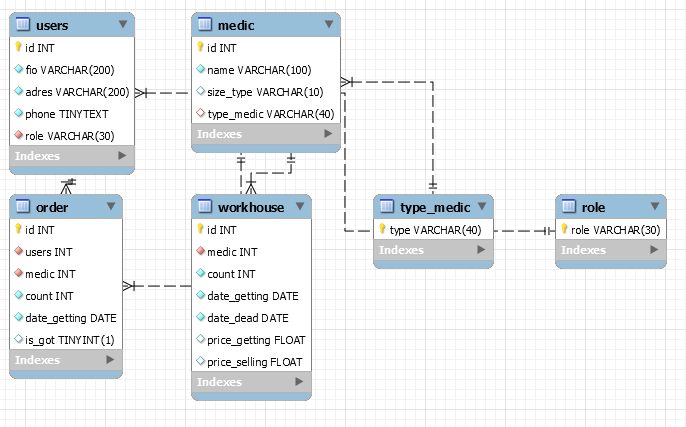
# Список использованных источников

1. 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
2. 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.
3. Гупта, Арун Java EE 7. Основы / Арун Гупта. - М. : Вильямс, 2014. - 336 c
4. Java и базы данных JDBC [Электронный ресурс]. – Режим доступа: https://metanit.com/java/database/1.1.php. – Дата доступа: 10.11.2020.
5. Java Swing [Электронный ресурс]. – Режим доступа: http://java-online.ru/libs-swing.xhtml. – Дата доступа: 10.11.2020.
6. MySql [Электронный ресурс]. – Режим доступа: https://www.methodlab.ru/technology/mysql.shtml. – Дата доступа: 10.10.2020.
7. Куликов, С. C. / Тестирование программного обеспечения. Базовый курс / С. С. Куликов. — Минск: Четыре четверти, 2017. — 312 с

# ПРИЛОЖЕНИЕ А – UML диаграмма



# ПРИЛОЖЕНИЕ Б – Схема базы данных



# ПРИЛОЖЕНИЕ В – Код программы

package tableModal;

import java.util.List;

import javax.swing.table.AbstractTableModel;

import entity.Warehouse;

/\*\*

\*

\*/

public class WarehouseTableModal extends AbstractTableModel {

public static final int ID\_COL = 0;

public static final int medic = 1;

public static final int count = 2;

public static final int dateGetting = 3;

public static final int dateDead = 4;

public static final int priceGetting = 5;

public static final int pricaeSelling = 6;

private final String[] columnNames = {"Id","Лекарство", "Количество", "Дата привоза","Годен до","Привоз по","Продажа по" };

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 medic:

return temp.getMedic();

case count:

return temp.getCount();

case dateGetting:

return temp.getDateGetting();

case dateDead:

return temp.getDateDead();

case priceGetting:

return temp.getPriceGetting();

case pricaeSelling:

return temp.getPricaeSelling();

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.Users;

/\*\*

\* модель таблицы клиентов

\* @author admin

\*

\*/

public class UsersTableModal extends AbstractTableModel {

private static final int ID\_COL = 0;

private static final int FIO\_COL = 1;

private static final int ADRES\_COL = 2;

private static final int PHONE\_COL = 3;

private static final int ROLE\_COL = 4;

private final String[] columnNames = {"ИД", "ФИО", "Адрес", "Телефон", "Роль" };

private List<Users> list;

public UsersTableModal(List<Users> 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) {

Users temp = list.get(row);

switch (col) {

case ID\_COL:

return temp.getId();

case FIO\_COL:

return temp.getFio();

case ADRES\_COL:

return temp.getAdres();

case PHONE\_COL:

return temp.getPhone();

case ROLE\_COL:

return temp.getRole();

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.Order;

/\*\*

\*

\* @author admin

\* модель таблицы заказа

\*/

public class OrderTableModal extends AbstractTableModel {

public static final int ID\_COL = 0;

public static final int USERS\_COL = 1;

public static final int MEDIC\_COL = 2;

public static final int COUNT\_COL = 3;

public static final int DATES\_COL = 4;

public static final int IS\_GOT\_COL = 5;

private final String[] columnNames = {"Id","Пользователь", "Лекарство", "Количество", "Дата доставки","Доставлено" };

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 USERS\_COL:

return temp.getUsers().getFio();

case MEDIC\_COL:

return temp.getMedic().getName();

case COUNT\_COL:

return temp.getCount();

case DATES\_COL:

return temp.getDate();

case IS\_GOT\_COL:

return temp.isGot();

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.MedicType;

/\*\*

\*

\*/

public class MedicTypeTableModal extends AbstractTableModel {

public static final int TYPE\_COL = 1;

private final String[] columnNames = {"Тип"};

private List<MedicType> list;

public MedicTypeTableModal(List<MedicType> 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) {

MedicType temp = list.get(row);

switch (col) {

case TYPE\_COL:

return temp.getType();

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.Medic;

import entity.Order;

/\*\*

\*

\*/

public class MedicTableModal extends AbstractTableModel {

public static final int ID\_COL = 0;

public static final int NAME\_COL = 1;

public static final int TYPE\_COL = 2;

public static final int SIZE\_TYPE\_COL = 3;

private final String[] columnNames = {"Id","Название", "Тип", "Объем" };

private List<Medic> list;

public MedicTableModal(List<Medic> 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) {

Medic temp = list.get(row);

switch (col) {

case ID\_COL:

return temp.getId();

case NAME\_COL:

return temp.getName();

case TYPE\_COL:

return temp.getType();

case SIZE\_TYPE\_COL:

return temp.getsizeType();

default:

return temp;

}

}

@Override

public Class getColumnClass(int c) {

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

}

} package forms;

import java.awt.BorderLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

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 controller.ApplicationController;

import entity.Medic;

import entity.MedicType;

import entity.Warehouse;

import com.toedter.calendar.JDateChooser;

import java.awt.FlowLayout;

import javax.swing.BoxLayout;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

public class WarehouseFrame extends JFrame{

private JTextField textField;

private JTextField textField\_1;

private JTextField textField\_2;

private JTable table;

JComboBox<Medic> comboBox;

private JTextField textField\_3;

private JDateChooser dateChooser;

private JDateChooser dateChooser\_1;

public WarehouseFrame() {

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

back();

}

});

setTitle("Склад");

setBounds(100, 100, 877, 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);

dateChooser = new JDateChooser();

panel\_1.setLayout(new BoxLayout(panel\_1, BoxLayout.Y\_AXIS));

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

panel\_1.add(lblNewLabel\_1\_1);

comboBox = new JComboBox();

panel\_1.add(comboBox);

JLabel lblNewLabel = new JLabel("Количество");

panel\_1.add(lblNewLabel);

textField\_1 = new JTextField();

textField\_1.addKeyListener(new KeyAdapter() {

@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();

}

}

});

textField\_1.setColumns(10);

panel\_1.add(textField\_1);

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

panel\_1.add(lblNewLabel\_2);

panel\_1.add(dateChooser);

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

btnNewButton\_3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionDeleteButton();

}

});

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();

}

});

JLabel lblNewLabel\_2\_1 = new JLabel("Истечение срока годности");

panel\_1.add(lblNewLabel\_2\_1);

dateChooser\_1 = new JDateChooser();

panel\_1.add(dateChooser\_1);

JLabel lblNewLabel\_3 = new JLabel("Цена поставки");

panel\_1.add(lblNewLabel\_3);

textField\_2 = new JTextField();

textField\_2.addKeyListener(new KeyAdapter() {

@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();

}

}

});

textField\_2.setColumns(10);

panel\_1.add(textField\_2);

JLabel lblNewLabel\_5 = new JLabel("Цена продажи");

panel\_1.add(lblNewLabel\_5);

textField\_3 = new JTextField();

textField\_3.addKeyListener(new KeyAdapter() {

@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();

}

}

});

textField\_3.setColumns(10);

panel\_1.add(textField\_3);

panel\_1.add(btnNewButton);

panel\_1.add(btnNewButton\_2);

panel\_1.add(btnNewButton\_3);

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();

Warehouse wh = (Warehouse) table.getModel().getValueAt(row, 99);

ApplicationController.warehouseController.actionDeleteButton(wh.getId());

}

}

private void actionTableMouseClicked() {

if (table.getRowCount() > 0)

try {

int row = table.getSelectedRow();

Warehouse wh = (Warehouse) table.getModel().getValueAt(row, 99);

comboBox.getModel().setSelectedItem(wh.getMedic());

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

Date date = Date.valueOf(wh.getDateGetting().toString());

dateChooser.setDate(date);

date = Date.valueOf(wh.getDateDead().toString());

dateChooser\_1.setDate(date);

textField\_2.setText(wh.getPriceGetting()+"");

textField\_3.setText(wh.getPricaeSelling()+"");

} catch (Exception e) {

}

}

private void actionUpdateButton() {

int column = 0;

int row = table.getSelectedRow();

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

ApplicationController.warehouseController.actionUpdateButton(

((Medic)comboBox.getSelectedItem()).getId(),

Integer.parseInt(textField\_1.getText()),

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

new Date(dateChooser\_1.getDate().getTime()),

Float.parseFloat(textField\_2.getText()),

Float.parseFloat(textField\_3.getText()),

id);

}

private void actionCreateButton() {

ApplicationController.warehouseController.actionCreateButton(((Medic)comboBox.getSelectedItem()).getId(),

Integer.parseInt(textField\_1.getText()),

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

new Date(dateChooser\_1.getDate().getTime()),

Float.parseFloat(textField\_2.getText()),

Float.parseFloat(textField\_3.getText()));

}

private void actionSearchButton() {

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

refreshComboBoxes();

}

public void refreshView() {

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

refreshComboBoxes();

}

public void refreshComboBoxes() {

try {

comboBox.setModel(

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

System.out.println("Перезагрузка списка");

} catch (Exception e) {

e.printStackTrace();

}

}

public void back() {

ApplicationController.warehouseController.back();

}

} package forms;

import java.awt.BorderLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

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 controller.ApplicationController;

import entity.Medic;

import entity.MedicType;

import entity.Users;

import javax.swing.BoxLayout;

/\*\*

\* форма отображения

\* @author admin

\*

\*/

public class UsersFrame extends JFrame {

private JTextField textField;

private JTextField textField\_1;

private JTextField textField\_2;

private JTable table;

private JTextField textField\_3;

public UsersFrame() {

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

back();

}

});

setTitle("Пользователи");

setBounds(100, 100, 776, 303);

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("ФИО");

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

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();

}

});

panel\_1.setLayout(new BoxLayout(panel\_1, BoxLayout.Y\_AXIS));

panel\_1.add(lblNewLabel);

textField\_1 = new JTextField();

textField\_1.setColumns(10);

panel\_1.add(textField\_1);

panel\_1.add(lblNewLabel\_1);

textField\_2 = new JTextField();

textField\_2.setColumns(10);

panel\_1.add(textField\_2);

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

panel\_1.add(lblNewLabel\_1\_2);

textField\_3 = new JTextField();

textField\_3.setColumns(10);

panel\_1.add(textField\_3);

panel\_1.add(btnNewButton);

panel\_1.add(btnNewButton\_2);

panel\_1.add(btnNewButton\_3);

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.UsersController.actionDeleteButton(Long.parseLong(id));

}

}

private void actionTableMouseClicked() {

if (table.getRowCount() > 0)

try {

int row = table.getSelectedRow();

Users user= (Users) table.getModel().getValueAt(row, 99);

textField\_1.setText(user.getFio());

textField\_2.setText(user.getAdres());

textField\_3.setText(user.getPhone());

} catch (Exception e) {

e.printStackTrace();

}

}

private void actionUpdateButton() {

int column = 0;

int row = table.getSelectedRow();

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

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

}

private void actionCreateButton() {

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

}

private void actionSearchButton() {

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

}

public void refreshView() {

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

}

public void back() {

ApplicationController.UsersController.back();

}

} package forms;

import java.awt.BorderLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

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 controller.ApplicationController;

import entity.Medic;

import entity.MedicType;

import entity.Warehouse;

import com.toedter.calendar.JDateChooser;

import java.awt.FlowLayout;

import javax.swing.BoxLayout;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

public class UserPersonalOrderFrame extends JFrame{

private JTextField textField;

private JTable table;

public UserPersonalOrderFrame() {

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

back();

}

});

setTitle("Мои заказы");

setBounds(100, 100, 742, 428);

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

JPanel panel = new JPanel();

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

JLabel lblNewLabel\_4 = new JLabel("Введите название лекарства 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.UserPersonalOrderController.actionSearchUsersOrderButton(textField.getText().trim(),true, table);

}

public void refreshView() {

ApplicationController.UserPersonalOrderController.actionSearchUsersOrderButton("".trim(),true, table);

}

public void back() {

ApplicationController.UserPersonalOrderController.back();

}

} package forms;

import java.awt.BorderLayout;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.sql.Date;

import javax.swing.DefaultComboBoxModel;

import javax.swing.JButton;

import javax.swing.JCheckBox;

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.border.EmptyBorder;

import com.toedter.calendar.JDateChooser;

import controller.ApplicationController;

import controller.UserOrderController;

import entity.Medic;

import entity.Order;

import entity.Users;

/\*\*

\* @author admin

\*

\*/

public class UserOrderFrame extends JFrame {

private JTextField textField;

private JTextField textField\_1;

private JTable table;

private JComboBox<Medic> comboBox\_1;

private JDateChooser dateChooser;

public UserOrderFrame() {

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

back();

}

});

setTitle("Заказы");

setBounds(100, 100, 823, 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);

textField\_1 = new JTextField();

textField\_1.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();

}

});

panel\_1.setLayout(new GridLayout(0, 1, 0, 0));

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

panel\_1.add(lblNewLabel\_1\_1);

comboBox\_1 = new JComboBox();

panel\_1.add(comboBox\_1);

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

panel\_1.add(lblNewLabel\_1\_2);

panel\_1.add(textField\_1);

JLabel lblNewLabel\_1\_2\_1 = new JLabel("Дата доставки");

panel\_1.add(lblNewLabel\_1\_2\_1);

dateChooser = new JDateChooser();

panel\_1.add(dateChooser);

panel\_1.add(btnNewButton);

panel\_1.add(btnNewButton\_2);

panel\_1.add(btnNewButton\_3);

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

refreshView();

}

}

private void actionTableMouseClicked() {

if (table.getRowCount() > 0)

try {

int row = table.getSelectedRow();

Order entity = (Order) table.getModel().getValueAt(row, 99);

Medic medic= ApplicationController.MedicController.getDAO().read(entity.getMedic().getId()).get(0);

comboBox\_1.getModel().setSelectedItem(medic);

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

dateChooser.setDate(entity.getDate());

} catch (Exception e) {

}

}

private void actionUpdateButton() {

int column = 0;

int row = table.getSelectedRow();

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

ApplicationController.UserOrderController.actionUpdateButton(id,UserOrderController.user,(Medic)comboBox\_1.getSelectedItem(),Integer.parseInt(textField\_1.getText()),

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

refreshView();

}

private void actionCreateButton() {

ApplicationController.orderController.actionCreateButton(UserOrderController.user,(Medic)comboBox\_1.getSelectedItem(),Integer.parseInt(textField\_1.getText()),

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

refreshView();

}

private void actionSearchButton() {

ApplicationController.orderController.actionSearchUsersOrderButton(textField.getText().trim(),false, table);

refreshComboBoxes();

}

public void refreshView() {

ApplicationController.orderController.actionSearchUsersOrderButton("",false, table);

refreshComboBoxes();

}

public void refreshComboBoxes() {

try {

comboBox\_1.setModel(

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

} catch (Exception e) {

e.printStackTrace();

}

}

public void back() {

ApplicationController.UserOrderController.back();

}

} package forms;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JTextField;

import controller.ApplicationController;

public class RegisterForm extends JFrame{

private JTextField textField;

private JTextField textField\_1;

private JTextField textField\_2;

/\*\*

\* Create the application.

\*/

public RegisterForm() {

setTitle("Регистрация");

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

setBounds(100, 100, 298, 240);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getContentPane().setLayout(null);

JLabel lblNewLabel = new JLabel("ФИО");

lblNewLabel.setBounds(10, 11, 46, 14);

getContentPane().add(lblNewLabel);

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

lblNewLabel\_1.setBounds(10, 49, 46, 14);

getContentPane().add(lblNewLabel\_1);

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

lblNewLabel\_2.setBounds(10, 93, 60, 14);

getContentPane().add(lblNewLabel\_2);

textField = new JTextField();

textField.setBounds(88, 8, 173, 20);

getContentPane().add(textField);

textField.setColumns(10);

textField\_1 = new JTextField();

textField\_1.setColumns(10);

textField\_1.setBounds(88, 46, 173, 20);

getContentPane().add(textField\_1);

textField\_2 = new JTextField();

textField\_2.setColumns(10);

textField\_2.setBounds(88, 90, 173, 20);

getContentPane().add(textField\_2);

JButton btnNewButton = new JButton("Регистрация");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btnRegisterAction();

}

});

btnNewButton.setBounds(10, 127, 251, 23);

getContentPane().add(btnNewButton);

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

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btnBackAction();

}

});

btnNewButton\_1.setBounds(10, 161, 251, 23);

getContentPane().add(btnNewButton\_1);

}

private void btnRegisterAction() {

if(textField.getText().length()>0 &&

textField\_1.getText().length()>0 &&

textField\_2.getText().length()>0 )

ApplicationController.RegistrationController.btnRegisterAction(textField.getText(),textField\_1.getText(),textField\_2.getText());

}

private void btnBackAction() {

ApplicationController.RegistrationController.btnBackAction();

}

} package forms;

import java.awt.BorderLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.sql.Date;

import java.util.List;

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 controller.ApplicationController;

import entity.Medic;

import entity.MedicType;

import entity.Order;

import entity.Users;

import java.awt.GridLayout;

import com.toedter.calendar.JDateChooser;

import javax.swing.JCheckBox;

/\*\*

\* форма отображения продуктов

\* @author admin

\*

\*/

public class OrderFrame extends JFrame {

private JTextField textField;

private JTextField textField\_1;

private JTable table;

private JComboBox<Users> comboBox;

private JComboBox<Medic> comboBox\_1;

private JDateChooser dateChooser;

private JCheckBox chckbxNewCheckBox;

public OrderFrame() {

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

back();

}

});

setTitle("Заказы");

setBounds(100, 100, 823, 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("Пользователь");

textField\_1 = new JTextField();

textField\_1.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();

}

});

panel\_1.setLayout(new GridLayout(0, 1, 0, 0));

panel\_1.add(lblNewLabel);

comboBox = new JComboBox();

panel\_1.add(comboBox);

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

panel\_1.add(lblNewLabel\_1\_1);

comboBox\_1 = new JComboBox();

panel\_1.add(comboBox\_1);

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

panel\_1.add(lblNewLabel\_1\_2);

panel\_1.add(textField\_1);

JLabel lblNewLabel\_1\_2\_1 = new JLabel("Дата доставки");

panel\_1.add(lblNewLabel\_1\_2\_1);

dateChooser = new JDateChooser();

panel\_1.add(dateChooser);

chckbxNewCheckBox = new JCheckBox("Доставлено?");

panel\_1.add(chckbxNewCheckBox);

panel\_1.add(btnNewButton);

panel\_1.add(btnNewButton\_2);

panel\_1.add(btnNewButton\_3);

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

}

}

private void actionTableMouseClicked() {

if (table.getRowCount() > 0)

try {

int row = table.getSelectedRow();

Order entity = (Order) table.getModel().getValueAt(row, 99);

Users users = ApplicationController.UsersController.getDAO().read(entity.getUsers().getId()).get(0);

Medic medic= ApplicationController.MedicController.getDAO().read(entity.getMedic().getId()).get(0);

comboBox.getModel().setSelectedItem(users);

comboBox\_1.getModel().setSelectedItem(medic);

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

dateChooser.setDate(entity.getDate());

chckbxNewCheckBox.enable(entity.isGot());

} catch (Exception e) {

}

}

private void actionUpdateButton() {

int column = 0;

int row = table.getSelectedRow();

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

ApplicationController.orderController.actionUpdateButton(id,(Users)comboBox.getSelectedItem(),(Medic)comboBox\_1.getSelectedItem(),Integer.parseInt(textField\_1.getText()),

new Date(dateChooser.getDate().getTime()),chckbxNewCheckBox.isSelected());

}

private void actionCreateButton() {

ApplicationController.orderController.actionCreateButton((Users)comboBox.getSelectedItem(),(Medic)comboBox\_1.getSelectedItem(),Integer.parseInt(textField\_1.getText()),

new Date(dateChooser.getDate().getTime()),chckbxNewCheckBox.isSelected() );

}

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.UsersController.getDAO().readAll().toArray()));

comboBox\_1.setModel(

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

} catch (Exception e) {

e.printStackTrace();

}

}

public void back() {

ApplicationController.orderController.back();

}

} 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.WindowAdapter;

import java.awt.event.WindowEvent;

/\*\*

\* форма отображения поставщиков

\*

\* @author admin

\*

\*/

public class MedicTypeFrame extends JFrame {

private JTable table;

private JTextField textField;

private JTextField textField\_4;

public MedicTypeFrame() {

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

back();

}

});

setTitle("Типы лекарств");

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 тип 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("Тип");

textField = new JTextField();

textField.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, false)

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

.addGap(4)

.addComponent(lblNewLabel))

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

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

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

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

.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(18)

.addComponent(btnNewButton)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(btnNewButton\_2)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(btnNewButton\_3)

.addGap(211))

);

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

ApplicationController.MedicTypeController.actionDeleteButton(type);

}

}

private void actionTableMouseClicked() {

if (table.getRowCount() > 0)

try {

int row = table.getSelectedRow();

textField.setText(table.getModel().getValueAt(row, 0).toString());

} catch (Exception e) {

}

}

private void actionUpdateButton() {

int column = 0;

int row = table.getSelectedRow();

ApplicationController.MedicTypeController.actionUpdateButton(textField.getText(),table.getModel().getValueAt(row, 0).toString());

refreshView();

}

private void actionCreateButton() {

ApplicationController.MedicTypeController.actionCreateButton(textField.getText());

}

private void actionSearchButton() {

ApplicationController.MedicTypeController.actionSearchButton(textField\_4.getText().trim(),

table);

}

public void refreshView() {

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

}

public void back() {

ApplicationController.MedicTypeController.back();;

}

} 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 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 controller.ApplicationController;

import entity.Medic;

import entity.MedicType;

import entity.Order;

import tableModal.MedicTableModal;

import tableModal.OrderTableModal;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

/\*\*

\* форма отображения продуктов

\* @author admin

\*

\*/

public class MedicFrame extends JFrame {

private JTextField textField;

private JTextField textField\_1;

private JTextField textField\_2;

private JTable table;

JComboBox<MedicType> comboBox;

public MedicFrame() {

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

back();

}

});

setTitle("Лекарства");

setBounds(100, 100, 803, 346);

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("В чем измеряется");

textField\_1 = new JTextField();

textField\_1.setColumns(10);

textField\_2 = new JTextField();

textField\_2.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();

}

});

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

comboBox = 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(gl\_panel\_1.createSequentialGroup()

.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(71, Short.MAX\_VALUE))

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

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

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

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

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

.addGap(71))

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

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

.addContainerGap(71, Short.MAX\_VALUE))

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

.addComponent(lblNewLabel\_1\_1, GroupLayout.PREFERRED\_SIZE, 88, GroupLayout.PREFERRED\_SIZE)

.addContainerGap(113, 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(18)

.addComponent(lblNewLabel\_1\_1)

.addGap(9)

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

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(btnNewButton)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(btnNewButton\_2)

.addPreferredGap(ComponentPlacement.RELATED)

.addComponent(btnNewButton\_3)

.addGap(106))

);

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.MedicController.actionDeleteButton(Long.parseLong(id));

}

}

private void actionTableMouseClicked() {

if (table.getRowCount() > 0)

try {

int row = table.getSelectedRow();

Medic medic = (Medic) table.getModel().getValueAt(row, 99);

textField\_1.setText(medic.getName());

comboBox.getModel().setSelectedItem(medic.getType());

textField\_2.setText(medic.getsizeType());

} catch (Exception e) {

}

}

private void actionUpdateButton() {

int column = 0;

int row = table.getSelectedRow();

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

ApplicationController.MedicController.actionUpdateButton(id,textField\_1.getText().trim(),textField\_2.getText().trim(), ((MedicType)comboBox.getSelectedItem()).getType());

}

private void actionCreateButton() {

ApplicationController.MedicController.actionCreateButton(textField\_1.getText().trim(),textField\_2.getText().trim(), ((MedicType)comboBox.getSelectedItem()).getType());

}

private void actionSearchButton() {

refreshComboBoxes();

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

}

public void refreshView() {

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

refreshComboBoxes();

}

public void refreshComboBoxes() {

try {

System.out.println("Перезагрузка списка");

comboBox.setModel(

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

} catch (Exception e) {

e.printStackTrace();

}

}

public void back() {

ApplicationController.MedicController.back();

}

} 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.SpinnerNumberModel;

import javax.swing.border.EmptyBorder;

import controller.ApplicationController;

import javax.swing.JSpinner;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

/\*\*

\* форма отображения склада

\* @author admin

\*

\*/

public class MedicDeadDate extends JFrame {

private JTable table;

private JSpinner spinner;

private SpinnerNumberModel model;

public MedicDeadDate() {

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

back();

}

});

setTitle("Лекарства с истекажщим сроком годности");

setBounds(100, 100, 835, 244);

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

JPanel panel = new JPanel();

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

JLabel lblNewLabel\_4 = new JLabel("Показать товары, срок годности которых истекает ранее или через количество дней");

panel.add(lblNewLabel\_4);

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

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

actionSearchButton();

}

});

model = new SpinnerNumberModel(30,1,720,1);

spinner = new JSpinner(model);

panel.add(spinner);

panel.add(btnNewButton\_1);

table = new JTable();

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

}

private void actionSearchButton() {

ApplicationController.warehouseController.searchByDeadDateDay(model.getNumber().intValue(), table);

}

public void refreshView() {

ApplicationController.warehouseController.searchByDeadDateDay(30, table);

}

public void back() {

ApplicationController.MedicDateDeadController.back();

}

} package forms;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JFrame;

import controller.ApplicationController;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

public class MainUsersFrame extends JFrame{

/\*\*

\* Create the application.

\*/

public MainUsersFrame() {

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

ApplicationController.mainUsersController.btnBackAction();

}

});

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

setBounds(100, 100, 233, 147);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getContentPane().setLayout(null);

JButton btnNewButton = new JButton("Новый заказ");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btnNewOrderAction();

}

});

btnNewButton.setBounds(10, 11, 186, 23);

getContentPane().add(btnNewButton);

JButton btnNewButton\_1 = new JButton("Мои заказы");

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btnMyOrdersAction();

}

});

btnNewButton\_1.setBounds(10, 45, 186, 23);

getContentPane().add(btnNewButton\_1);

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

btnNewButton\_1\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btnBackAction();

}

});

btnNewButton\_1\_1.setBounds(10, 74, 186, 23);

getContentPane().add(btnNewButton\_1\_1);

}

private void btnNewOrderAction() {

ApplicationController.mainUsersController.btnNewOrderAction();

}

private void btnMyOrdersAction() {

ApplicationController.mainUsersController.btnMyOrdersAction();

}

private void btnBackAction() {

ApplicationController.mainUsersController.btnBackAction();

}

} package forms;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JFrame;

import controller.ApplicationController;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

/\*\*

\* форма главного меню

\* @author admin

\*

\*/

public class MainAdminFrame extends JFrame{

public MainAdminFrame() {

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

back();

}

});

initialize();

}

private void initialize() {

setTitle("Главная");

setBounds(100, 100, 233, 257);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getContentPane().setLayout(null);

JButton btnNewButton = new JButton("Типы лекарств");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btnMedicTypeAction();

}

});

btnNewButton.setBounds(10, 11, 197, 23);

getContentPane().add(btnNewButton);

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

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btnMedicAction();

}

});

btnNewButton\_1.setBounds(10, 45, 197, 23);

getContentPane().add(btnNewButton\_1);

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

btnNewButton\_2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btnWarehouseAction();

}

});

btnNewButton\_2.setBounds(10, 79, 197, 23);

getContentPane().add(btnNewButton\_2);

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

btnNewButton\_3.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btnUsersAction();

}

});

btnNewButton\_3.setBounds(10, 113, 197, 23);

getContentPane().add(btnNewButton\_3);

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

btnNewButton\_4.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btnOrderAction();

}

});

btnNewButton\_4.setBounds(10, 147, 197, 23);

getContentPane().add(btnNewButton\_4);

JButton btnNewButton\_5 = new JButton("Истекает срок");

btnNewButton\_5.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btnDateDeadAction();

}

});

btnNewButton\_5.setBounds(10, 181, 197, 23);

getContentPane().add(btnNewButton\_5);

}

private void btnMedicTypeAction() {

ApplicationController.mainAdminController.btnMedicTypeAction();

}

private void btnMedicAction() {

ApplicationController.mainAdminController.btnMedicAction();

}

private void btnWarehouseAction() {

ApplicationController.mainAdminController.btnWarehouseAction();

}

private void btnUsersAction() {

ApplicationController.mainAdminController.btnUsersAction();

}

private void btnOrderAction() {

ApplicationController.mainAdminController.btnOrderAction();

}

private void btnDateDeadAction() {

ApplicationController.mainAdminController.btnDateDeadAction();

}

public void back() {

ApplicationController.mainAdminController.back();

}

} package forms;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JTextField;

import controller.ApplicationController;

public class LoginFrame extends JFrame{

private JTextField textField;

private JTextField textField\_1;

/\*\*

\* Create the application.

\*/

public LoginFrame() {

setTitle("Вход");

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

setBounds(100, 100, 433, 215);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

getContentPane().setLayout(null);

JLabel lblNewLabel = new JLabel("Логин (ФИО)");

lblNewLabel.setBounds(10, 11, 157, 14);

getContentPane().add(lblNewLabel);

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

lblNewLabel\_1.setBounds(10, 50, 157, 14);

getContentPane().add(lblNewLabel\_1);

textField = new JTextField();

textField.setBounds(177, 8, 231, 20);

getContentPane().add(textField);

textField.setColumns(10);

textField\_1 = new JTextField();

textField\_1.setColumns(10);

textField\_1.setBounds(177, 47, 231, 20);

getContentPane().add(textField\_1);

JButton btnNewButton = new JButton("Войти");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btnLoginAction();

}

});

btnNewButton.setBounds(10, 78, 398, 23);

getContentPane().add(btnNewButton);

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

btnNewButton\_1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btnRegisterAction();

}

});

btnNewButton\_1.setBounds(10, 112, 398, 23);

getContentPane().add(btnNewButton\_1);

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

btnNewButton\_2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

btnExitAction();

}

});

btnNewButton\_2.setBounds(10, 148, 398, 23);

getContentPane().add(btnNewButton\_2);

}

private void btnLoginAction() {

if(textField.getText().length()>0 &&textField\_1.getText().length()>0 )

ApplicationController.LoginController.btnLoginAction(textField.getText(),textField\_1.getText());

}

private void btnRegisterAction() {

ApplicationController.LoginController.btnRegisterAction();

}

private void btnExitAction() {

System.exit(0);

}

} package entity;

import java.sql.Date;

public class Warehouse extends AbstractEntity{

private Medic medic;

private int count;

private Date dateGetting;

private Date dateDead;

private float priceGetting;

private float pricaeSelling;

public Warehouse() {

super();

// TODO Auto-generated constructor stub

}

public Warehouse(Medic medic, int count, Date dateGetting, Date dateDead, float priceGetting, float pricaeSelling) {

super();

this.medic = medic;

this.count = count;

this.dateGetting = dateGetting;

this.dateDead = dateDead;

this.priceGetting = priceGetting;

this.pricaeSelling = pricaeSelling;

}

public Medic getMedic() {

return medic;

}

public void setMedic(Medic medic) {

this.medic = medic;

}

public int getCount() {

return count;

}

public void setCount(int count) {

this.count = count;

}

public Date getDateGetting() {

return dateGetting;

}

public void setDateGetting(Date dateGetting) {

this.dateGetting = dateGetting;

}

public Date getDateDead() {

return dateDead;

}

public void setDateDead(Date dateDead) {

this.dateDead = dateDead;

}

public float getPriceGetting() {

return priceGetting;

}

public void setPriceGetting(float priceGetting) {

this.priceGetting = priceGetting;

}

public float getPricaeSelling() {

return pricaeSelling;

}

public void setPricaeSelling(float pricaeSelling) {

this.pricaeSelling = pricaeSelling;

}

@Override

public String toString() {

return "medic=" + medic + ", count=" + count + ", dateGetting=" + dateGetting + ", dateDead="

+ dateDead + ", priceGetting=" + priceGetting + ", pricaeSelling=" + pricaeSelling;

}

}

package entity;

public class Users extends AbstractEntity {

private String fio;

private String adres;

private String phone;

private String role;

public Users() {

super();

}

public Users(String fio, String adres, String phone, String role) {

super();

this.fio = fio;

this.adres = adres;

this.phone = phone;

this.role = role;

}

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 adres;

}

public void setEmail(String email) {

this.adres = email;

}

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

public String getAdres() {

return adres;

}

public void setAdres(String adres) {

this.adres = adres;

}

@Override

public String toString() {

return fio;

}

}

package entity;

public class Role extends AbstractEntity{

private String role;

public String getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

public Role(String role) {

super();

this.role = role;

}

public Role() {

super();

// TODO Auto-generated constructor stub

}

@Override

public String toString() {

return role;

}

}

package entity;

import java.sql.Date;

public class Order extends AbstractEntity{

private Users users;

private Medic medic;

private int count;

private Date date;

private boolean isGot;

public Order() {

super();

// TODO Auto-generated constructor stub

}

public Order(Users users, Medic medic, int count, Date date, boolean isGot) {

super();

this.users = users;

this.medic = medic;

this.count = count;

this.date = date;

this.isGot = isGot;

}

public Users getUsers() {

return users;

}

public void setUsers(Users users) {

this.users = users;

}

public Medic getMedic() {

return medic;

}

public void setMedic(Medic medic) {

this.medic = medic;

}

public int getCount() {

return count;

}

public void setCount(int count) {

this.count = count;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public boolean isGot() {

return isGot;

}

public void setGot(boolean isGot) {

this.isGot = isGot;

}

@Override

public String toString() {

return "users=" + users + ", medic=" + medic + ", count=" + count + ", date=" + date + ", isGot=" + isGot;

}

}

package entity;

public class MedicType extends AbstractEntity{

private String type;

public String getType() {

return type;

}

public void setType(String type) {

this.type = type;

}

public MedicType(String type) {

super();

this.type = type;

}

public MedicType() {

super();

// TODO Auto-generated constructor stub

}

@Override

public String toString() {

return type;

}

}

package entity;

public class Medic extends AbstractEntity{

private String name;

private MedicType type;

private String sizeType;

public Medic() {

super();

// TODO Auto-generated constructor stub

}

public Medic(String name, MedicType type, String sizeType) {

super();

this.name = name;

this.type = type;

this.sizeType = sizeType;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public MedicType getType() {

return type;

}

public void setType(MedicType type) {

this.type = type;

}

public String getsizeType() {

return sizeType;

}

public void setsizeType(String sizeType) {

this.sizeType = sizeType;

}

@Override

public String toString() {

return name + ", " + type;

}

}

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;

}

}

package 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 defaultOperation;

import java.util.List;

public interface CrudOperation<T>{

boolean Create(T entity);

List<T> Read();

T Read(Long id);

boolean Update(T entity);

boolean Delete(T entity);

boolean Delete(Long id);

}

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.Medic;

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 workhouse 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 workhouse");

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 workhouse 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 workhouse WHERE medic = ?");

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 workhouse WHERE medic in (SELECT id FROM medic 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 List<Warehouse > searchByDeadDateDay(int day) throws Exception {

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

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.prepareStatement("SELECT \* FROM workhouse WHERE DATE\_ADD(date\_getting, INTERVAL ? DAY) > date\_dead");

myStmt.setInt(1, day);

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 workhouse"

+ " (medic, count,date\_getting,date\_dead,price\_getting, price\_selling)" + " values ( ?, ?, ?, ?, ?, ?)");

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

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

myStmt.setDate(3, entity.getDateGetting());

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

myStmt.setFloat(5, entity.getPriceGetting());

myStmt.setFloat(6, entity.getPricaeSelling());

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 workhouse 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 workhouse SET medic=?, count=? , date\_getting=?, date\_dead=?,price\_getting=?, price\_selling=? WHERE id=?");

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

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

myStmt.setDate(3, entity.getDateGetting());

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

myStmt.setFloat(5, entity.getPriceGetting());

myStmt.setFloat(6, entity.getPricaeSelling());

myStmt.setLong(7, entity.getId());

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

public void Delete(Long id) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("DELETE FROM workhouse 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\_medic = myRs.getLong("medic");

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

Date dategetting = myRs.getDate("date\_getting");

Date datedead = myRs.getDate("date\_dead");

Float pricegetting = myRs.getFloat("price\_getting");

Float priceselling = myRs.getFloat("price\_selling");

Medic medic = null;

try {

medic = ApplicationController.MedicController.getDAO().read(id\_medic).get(0);

} catch (Exception e) {

e.printStackTrace();

}

Warehouse temp = new Warehouse(medic, count, dategetting, datedead, pricegetting, priceselling);

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.Users;

public class UserDAO {

private Connection myConn;

public UserDAO() 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 USER connection success");

}

public List<Users > readAll() throws Exception {

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

Statement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.createStatement();

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

while (myRs.next()) {

Users tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public List<Users > searchByFIO(String name) throws Exception {

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

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

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

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

myStmt.setString(1, name);

myRs = myStmt.executeQuery();

while (myRs.next()) {

Users tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public Users searchByFIOAndPhone(String fio, String phone) throws Exception {

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

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.prepareStatement("SELECT \* FROM Users WHERE fio=? AND phone=?");

myStmt.setString(1, fio);

myStmt.setString(2, phone);

myRs = myStmt.executeQuery();

while (myRs.next()) {

Users tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list.size()==0?null:list.get(0);

} finally {

close(myStmt, myRs);

}

}

public void create(Users entity) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn

.prepareStatement("insert into Users" + " (fio, adres, phone, role)" + " values (?, ?,?,?)");

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

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

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

myStmt.setString(4, entity.getRole());

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

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

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

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn

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

myStmt.setLong(1, id);

myRs=myStmt.executeQuery();

while (myRs.next()) {

Users tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void update(Users entity) throws Exception {

PreparedStatement myStmt = null;

try {

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

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

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

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

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 Users WHERE id=?");

myStmt.setLong(1, id);

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

private Users convertRowToEntity(ResultSet myRs) throws SQLException {

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

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

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

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

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

Users temp = new Users (fio, adres, phone, role);

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 controller.UserOrderController;

import entity.Medic;

import entity.Order;

import entity.Users;

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 ORDER 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 medicinedeliverysimulator.order");

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 medicinedeliverysimulator.order WHERE users in (SELECT id from users where fio 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 medicinedeliverysimulator.order"

+ " (users, medic, count, date\_getting, is\_got)" + " values (?, ?, ?, ?,?)");

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

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

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

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

myStmt.setBoolean(5, entity.isGot());

myStmt.executeUpdate();

} 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 medicinedeliverysimulator.order 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 {

myStmt = myConn.prepareStatement("UPDATE medicinedeliverysimulator.order SET users=?, medic=?, count=?, date\_getting=?,is\_got=? WHERE id=?");

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

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

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

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

myStmt.setBoolean(5, entity.isGot());

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

myStmt.executeUpdate();

} 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 medicinedeliverysimulator.order WHERE id=?");

myStmt.setLong(1, id);

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

private Order convertRowToEntity(ResultSet myRs) throws SQLException {

Medic medic = null;

Users users = null;

Order temp = null;

try {

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

Long id\_users = myRs.getLong("users");

Long id\_medic = myRs.getLong("medic");

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

Date dategetting = myRs.getDate("date\_getting");

Boolean isGot = myRs.getBoolean("is\_got");

users = ApplicationController.UsersController.getDAO().read(id\_users).get(0);

medic = ApplicationController.MedicController.getDAO().read(id\_medic).get(0);

temp = new Order(users, medic,count, dategetting, isGot);

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);

}

public List<Order> searchMyUserAndMedic(String medic,boolean isGot) throws SQLException {

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

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

medic = "%" + medic + "%";

myStmt = myConn.prepareStatement(

"SELECT \* FROM medicinedeliverysimulator.order WHERE is\_got=? and users=? and medic in (SELECT id from medic where name LIKE ?)");

myStmt.setBoolean(1, isGot);

myStmt.setLong(2, UserOrderController.user.getId());

myStmt.setString(3, medic);

myRs = myStmt.executeQuery();

while (myRs.next()) {

Order tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

} 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.MedicType;

public class MedicTypeDAO {

private Connection myConn;

public MedicTypeDAO() 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 medicType connection success");

}

public List<MedicType> readAll() throws Exception {

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

Statement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.createStatement();

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

while (myRs.next()) {

MedicType tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

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

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

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

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

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

myStmt.setString(1, name);

myRs = myStmt.executeQuery();

while (myRs.next()) {

MedicType tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void create(MedicType entity) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn

.prepareStatement("insert into type\_medic" + " (type)" + " values (?)");

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

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

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

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

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn

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

myStmt.setLong(1, id);

myRs=myStmt.executeQuery();

while (myRs.next()) {

MedicType tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void update(String typeNew, String typeOld) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("UPDATE type\_medic SET type=? WHERE type=?");

myStmt.setString(1, typeNew);

myStmt.setString(2, typeOld);

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

public void Delete(String type) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("DELETE FROM type\_medic WHERE type=?");

myStmt.setString(1, type);

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

private MedicType convertRowToEntity(ResultSet myRs) throws SQLException {

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

MedicType temp = new MedicType(type);

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 controller.ApplicationController;

import entity.Medic;

import entity.MedicType;

public class MedicDAO {

private Connection myConn;

public MedicDAO() 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 medic connection success");

}

public List<Medic > readAll() throws Exception {

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

Statement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn.createStatement();

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

while (myRs.next()) {

Medic tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

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

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

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

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

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

myStmt.setString(1, name);

myRs = myStmt.executeQuery();

while (myRs.next()) {

Medic tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void create(Medic entity) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn

.prepareStatement("insert into medic" + " (name, size\_type,type\_medic)" + " values (?, ?,?)");

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

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

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

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

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

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

PreparedStatement myStmt = null;

ResultSet myRs = null;

try {

myStmt = myConn

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

myStmt.setLong(1, id);

myRs=myStmt.executeQuery();

while (myRs.next()) {

Medic tempEntity = convertRowToEntity(myRs);

list.add(tempEntity);

}

return list;

} finally {

close(myStmt, myRs);

}

}

public void update(Medic entity) throws Exception {

PreparedStatement myStmt = null;

try {

myStmt = myConn.prepareStatement("UPDATE medic SET name=?, size\_type=?, type\_medic=? WHERE id=?");

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

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

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

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 medic WHERE id=?");

myStmt.setLong(1, id);

myStmt.executeUpdate();

} finally {

close(myStmt);

}

}

private Medic convertRowToEntity(ResultSet myRs) throws SQLException {

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

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

String size\_type= myRs.getString("size\_type");

String type\_medic= myRs.getString("type\_medic");

Medic temp = new Medic (name, new MedicType(type\_medic),size\_type);

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.sql.Date;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import dao.WarehouseDAO;

import defaultOperation.StandartFrameOperation;

import entity.Medic;

import entity.Warehouse;

import forms.WarehouseFrame;

import tableModal.MedicTableModal;

import tableModal.UsersTableModal;

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 medicname, JTable table) {

try {

List<Warehouse> list = null;

if (medicname.trim().length() > 0)

list = DAO.search(medicname);

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 searchByDeadDateDay(int day, JTable table) {

try {

List<Warehouse> list = null;

list = DAO.searchByDeadDateDay(day);

WarehouseTableModal model = new WarehouseTableModal(list);

table.setModel(model);

} catch (Exception e) {

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

}

}

public void actionCreateButton(Long medic, int count, Date dateGetting, Date dateDead, float priceGetting,

float pricaeSelling) {

try {

if (medic != null && count >= 0 && priceGetting > 0 && pricaeSelling > 0) {

Medic md = ApplicationController.MedicController.DAO.read(medic).get(0);

Warehouse entity = new Warehouse(md, count, dateGetting, dateDead, priceGetting, pricaeSelling);

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 medic, int count, Date dateGetting, Date dateDead, float priceGetting,

float pricaeSelling, Long id) {

try {

Medic md = ApplicationController.MedicController.DAO.read(medic).get(0);

Warehouse entity = new Warehouse(md, count, dateGetting, dateDead, priceGetting, pricaeSelling);

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);

}

}

@Override

public void switchVisible() {

super.switchVisible();

refrechView();

}

public void refrechView() {

((WarehouseFrame) getFrame()).refreshView();

}

public WarehouseDAO getDAO() {

return DAO;

}

public void back() {

switchVisible();

ApplicationController.mainAdminController.switchVisible();

}

} package controller;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import dao.UserDAO;

import defaultOperation.StandartFrameOperation;

import entity.Users;

import forms.MedicFrame;

import forms.UsersFrame;

import tableModal.UsersTableModal;

public class UsersController extends StandartFrameOperation{

UserDAO DAO;

public UsersController(JFrame frame) {

super(frame);

try {

DAO = new UserDAO();

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

}

public void actionSearchButton(String fio, JTable table) {

try {

List<Users> list = null;

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

list = DAO.searchByFIO(fio);

else

list = DAO.readAll();

UsersTableModal model = new UsersTableModal(list);

table.setModel(model);

} catch (Exception e) {

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

}

}

public Users login(String fio,String phone) {

Users user = null;

try {

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

user = DAO.searchByFIOAndPhone(fio,phone);

} catch (Exception e) {

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

}

return user;

}

public void actionCreateButton(String fio, String adres, String phone) {

if (fio.length() > 0 && phone.length() > 0 && adres.length() > 0) {

Users entity = new Users(fio, adres, phone,"user");

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 adres, String phone, Long id) {

if (fio.length() > 0 && phone.length() > 0 && adres.length() > 0 && id>0) {

try {

Users entity= new Users(fio, adres, phone,"user");

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);

}

}

@Override

public void switchVisible() {

super.switchVisible();

refrechView();

}

public void refrechView() {

((UsersFrame) getFrame()).refreshView();

}

public UserDAO getDAO() {return DAO;}

public void back() {

switchVisible();

ApplicationController.mainAdminController.switchVisible();

}

} package controller;

import java.util.ArrayList;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import dao.OrderDAO;

import defaultOperation.StandartFrameOperation;

import entity.Order;

import forms.UserPersonalOrderFrame;

import tableModal.OrderTableModal;

public class UserPersonalOrderController extends StandartFrameOperation {

OrderDAO DAO;

public UserPersonalOrderController(JFrame frame) {

super(frame);

try {

DAO = new OrderDAO();

} catch (Exception e) {

e.printStackTrace();

}

}

@Override

public void switchVisible() {

super.switchVisible();

refrechView();

}

public void refrechView() {

((UserPersonalOrderFrame) getFrame()).refreshView();

}

public OrderDAO getDAO() {

return DAO;

}

public void back() {

switchVisible();

ApplicationController.mainUsersController.switchVisible();

}

public void actionSearchUsersOrderButton(String medic, boolean isGot, JTable table) {

try {

List<Order> list = null;

list = DAO.searchMyUserAndMedic(medic,isGot);

if(list==null)list=new ArrayList<Order>();

OrderTableModal model = new OrderTableModal(list);

table.setModel(model);

} catch (Exception e) {

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

}

}

} package controller;

import java.sql.Date;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import dao.MedicDAO;

import dao.OrderDAO;

import defaultOperation.StandartFrameOperation;

import entity.Medic;

import entity.MedicType;

import entity.Order;

import entity.Users;

import forms.MedicFrame;

import forms.OrderFrame;

import forms.UserOrderFrame;

import tableModal.OrderTableModal;

public class UserOrderController extends StandartFrameOperation {

public static Users user;

OrderDAO DAO;

public UserOrderController(JFrame frame) {

super(frame);

try {

DAO = new OrderDAO();

} catch (Exception e) {

e.printStackTrace();

}

}

public void actionSearchButton(String userFIO, JTable table) {

try {

List<Order> list = null;

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

list = DAO.search(userFIO);

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(Users users, Medic medic, int count, Date date, boolean isGot) {

if (users!=null &&medic!=null && count > 0) {

Order entity = new Order ( users, medic, count, date, isGot);

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,Users users, Medic medic, int count, Date date, boolean isGot) {

if (users!=null &&medic!=null && count > 0) {

try {

Order entity = new Order ( users, medic, count, date, isGot);

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);

}

}

@Override

public void switchVisible() {

super.switchVisible();

refrechView();

}

public void refrechView() {

((UserOrderFrame) getFrame()).refreshView();

}

public OrderDAO getDAO() {

return DAO;

}

public void back() {

switchVisible();

ApplicationController.mainUsersController.switchVisible();

}

} package controller;

import javax.swing.JFrame;

import defaultOperation.StandartFrameOperation;

public class RegistrationController extends StandartFrameOperation {

public RegistrationController(JFrame frame) {

super(frame);

}

public void btnRegisterAction(String fio, String adres,String phone) {

ApplicationController.UsersController.actionCreateButton(fio, adres, phone);

}

public void btnBackAction() {

switchVisible();

ApplicationController.LoginController.switchVisible();

}

} package controller;

import java.sql.Date;

import java.util.ArrayList;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import dao.OrderDAO;

import defaultOperation.StandartFrameOperation;

import entity.Medic;

import entity.Order;

import entity.Users;

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 userFIO, JTable table) {

try {

List<Order> list = null;

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

list = DAO.search(userFIO);

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(Users users, Medic medic, int count, Date date, boolean isGot) {

if (users!=null &&medic!=null && count > 0) {

Order entity = new Order ( users, medic, count, date, isGot);

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,Users users, Medic medic, int count, Date date, boolean isGot) {

if (users!=null &&medic!=null && count > 0) {

try {

Order entity = new Order ( users, medic, count, date, isGot);

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);

}

}

@Override

public void switchVisible() {

super.switchVisible();

refrechView();

}

public void refrechView() {

((OrderFrame) getFrame()).refreshView();

}

public OrderDAO getDAO() {

return DAO;

}

public void back() {

switchVisible();

ApplicationController.mainAdminController.switchVisible();

}

public void actionSearchUsersOrderButton(String medic, boolean isGot, JTable table) {

try {

List<Order> list = null;

list = DAO.searchMyUserAndMedic(medic,isGot);

if(list==null)list=new ArrayList<Order>();

OrderTableModal model = new OrderTableModal(list);

table.setModel(model);

} catch (Exception e) {

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

}

}

} package controller;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import dao.MedicTypeDAO;

import defaultOperation.StandartFrameOperation;

import entity.MedicType;

import forms.MedicTypeFrame;

import tableModal.MedicTypeTableModal;

public class MedicTypeController extends StandartFrameOperation {

MedicTypeDAO DAO;

public MedicTypeController(JFrame frame) {

super(frame);

try {

DAO = new MedicTypeDAO();

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

}

public void actionSearchButton(String name, JTable table) {

try {

List<MedicType> list = null;

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

list = DAO.search(name);

else

list = DAO.readAll();

MedicTypeTableModal model = new MedicTypeTableModal(list);

table.setModel(model);

} catch (Exception e) {

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

}

}

public void actionCreateButton(String type) {

if (type.length() > 0 ) {

MedicType mt = new MedicType(type);

try {

DAO.create(mt);

((MedicTypeFrame) 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 typeNew, String typeOld) {

if (typeNew.length()>0 ) {

try {

MedicType mt = new MedicType(typeNew);

DAO.update(typeNew,typeOld);

((MedicTypeFrame) 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(String type) {

try {

DAO.Delete(type);

((MedicTypeFrame) 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 MedicTypeDAO getDAO() {

return DAO;

}

public void back() {

switchVisible();

ApplicationController.mainAdminController.switchVisible();

}

} package controller;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import dao.MedicTypeDAO;

import defaultOperation.StandartFrameOperation;

import entity.MedicType;

import forms.MedicTypeFrame;

import tableModal.MedicTypeTableModal;

public class MedicDateDeadController extends StandartFrameOperation {

MedicTypeDAO DAO;

public MedicDateDeadController(JFrame frame) {

super(frame);

try {

DAO = new MedicTypeDAO();

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

}

public MedicTypeDAO getDAO() {

return DAO;

}

public void back() {

switchVisible();

ApplicationController.mainAdminController.switchVisible();

}

} package controller;

import java.util.List;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JTable;

import dao.MedicDAO;

import defaultOperation.StandartFrameOperation;

import entity.Medic;

import entity.MedicType;

import forms.MedicFrame;

import tableModal.MedicTableModal;

public class MedicController extends StandartFrameOperation {

public MedicDAO DAO;

public MedicController(JFrame frame) {

super(frame);

try {

DAO = new MedicDAO();

} catch (Exception e) {

e.printStackTrace();

}

}

public void actionSearchButton(String name, JTable table) {

try {

List<Medic> list = null;

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

list = DAO.search(name);

else

list = DAO.readAll();

MedicTableModal model = new MedicTableModal(list);

table.setModel(model);

} catch (Exception e) {

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

}

}

public void actionCreateButton(String name, String sizetype, String medictype) {

if (name != null && name.length() > 0) {

Medic entity = new Medic(name, new MedicType(medictype), sizetype);

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, String name, String sizetype, String medictype) {

if (name != null) {

try {

Medic entity = new Medic(name, new MedicType(medictype), sizetype);

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);

}

}

@Override

public void switchVisible() {

super.switchVisible();

refrechView();

}

public void refrechView() {

((MedicFrame) getFrame()).refreshView();

}

public MedicDAO getDAO() {

return DAO;

}

public void back() {

switchVisible();

ApplicationController.mainAdminController.switchVisible();

}

} package controller;

import javax.swing.JFrame;

import defaultOperation.StandartFrameOperation;

public class MainUsersController extends StandartFrameOperation {

public MainUsersController(JFrame frame) {

super(frame);

}

public void btnNewOrderAction() {

switchVisible();

ApplicationController.UserOrderController.switchVisible();

}

public void btnMyOrdersAction() {

ApplicationController.UserPersonalOrderController.switchVisible();

switchVisible();

}

public void btnBackAction() {

switchVisible();

ApplicationController.LoginController.switchVisible();

}

} package controller;

import javax.swing.JFrame;

import defaultOperation.StandartFrameOperation;

public class MainController extends StandartFrameOperation {

public MainController(JFrame frame) {

super(frame);

}

// public void actionWarehouseButton() {

// ApplicationController.warehouseController.switchVisible();

// }

} package controller;

import javax.swing.JFrame;

import defaultOperation.StandartFrameOperation;

import forms.MedicDeadDate;

public class MainAdminController extends StandartFrameOperation {

public MainAdminController(JFrame frame) {

super(frame);

}

public void btnMedicTypeAction() {

switchVisible();

ApplicationController.MedicTypeController.switchVisible();

}

public void btnMedicAction() {

switchVisible();

ApplicationController.MedicController.switchVisible();

}

public void btnWarehouseAction() {

switchVisible();

ApplicationController.warehouseController.switchVisible();

}

public void btnUsersAction() {

switchVisible();

ApplicationController.UsersController.switchVisible();

}

public void btnOrderAction() {

switchVisible();

ApplicationController.orderController.switchVisible();

}

public void btnDateDeadAction() {

switchVisible();

ApplicationController.MedicDateDeadController.switchVisible();

}

public void back() {

switchVisible();

ApplicationController.LoginController.switchVisible();

}

} package controller;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import defaultOperation.StandartFrameOperation;

import entity.Users;

public class LoginController extends StandartFrameOperation {

public LoginController(JFrame frame) {

super(frame);

}

public void btnLoginAction(String fio, String phone) {

Users login = ApplicationController.UsersController.login(fio, phone);

if (login != null) {

if (login.getRole().equals("user")) {

switchVisible();

ApplicationController.UserOrderController.user=login;

ApplicationController.mainUsersController.switchVisible();

} else {

switchVisible();

ApplicationController.mainAdminController.switchVisible();

}

} else

JOptionPane.showMessageDialog(getFrame(), "Нет таких пользователей", "Не найдено",

JOptionPane.ERROR\_MESSAGE);

}

public void btnRegisterAction() {

ApplicationController.RegistrationController.switchVisible();

switchVisible();

}

} package controller;

import forms.LoginFrame;

import forms.MainAdminFrame;

import forms.MainUsersFrame;

import forms.MedicDeadDate;

import forms.MedicFrame;

import forms.MedicTypeFrame;

import forms.OrderFrame;

import forms.RegisterForm;

import forms.UserOrderFrame;

import forms.UserPersonalOrderFrame;

import forms.UsersFrame;

import forms.WarehouseFrame;

public class ApplicationController {

// страница входа

public static LoginController LoginController = new LoginController(new LoginFrame());

// страница регистрации

public static RegistrationController RegistrationController = new RegistrationController(new RegisterForm());

// страница админа

public static MainAdminController mainAdminController = new MainAdminController(new MainAdminFrame());

// страница пользователя

public static MainUsersController mainUsersController = new MainUsersController(new MainUsersFrame());

// страница лекарств

public static MedicController MedicController = new MedicController(new MedicFrame());

// страница типов лекарст

public static MedicTypeController MedicTypeController = new MedicTypeController(new MedicTypeFrame());

// страница заказов

public static OrderController orderController = new OrderController(new OrderFrame());

// страница заказов клиента

public static UserOrderController UserOrderController = new UserOrderController(new UserOrderFrame());

// страница таблицы пользователя

public static UsersController UsersController = new UsersController(new UsersFrame());

// страница склада

public static WarehouseController warehouseController = new WarehouseController(new WarehouseFrame());

//срок годности

public static MedicDateDeadController MedicDateDeadController = new MedicDateDeadController(new MedicDeadDate());

//личные заказы

public static UserPersonalOrderController UserPersonalOrderController = new UserPersonalOrderController (new UserPersonalOrderFrame() );

public static void main(String[] args) {

LoginController.showFrame();

}

}