Министерство образования Республики Беларусь

Учреждение образования Белорусский государственный университет информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Объектно-ориентированное программирование (ООП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

«Программа магазин для сбыта плодов и овощей»

Выполнила: студентка гр.953501 Голубович Ю.И.

Руководитель: ассистент кафедры информатики Рогов М.Г.

Минск 2021

Содержание

[Введение 3](#_Toc73543629)

[1. Постановка задачи 4](#_Toc73543630)

[1.1. Описание предметной области 4](#_Toc73543631)

[1.2. Требования к разрабатываемому программному обеспечению 6](#_Toc73543632)

[2. Проектирование задачи 7](#_Toc73543633)

[2.1. Проектирование архитектуры приложения 7](#_Toc73543634)

[2.2. Используемые технологии 8](#_Toc73543635)

[2.3. Среды реализации задачи 8](#_Toc73543636)

[2.3.1. IntelliJ IDEA 8](#_Toc73543637)

[2.3.2. Java 9](#_Toc73543638)

[2.3.3. JavaFX 10](#_Toc73543639)

[2.3.4. SceneBuilder 10](#_Toc73543640)

[3. Программная реализация 12](#_Toc73543641)

[3.1. Структура приложения 12](#_Toc73543642)

[3.2 Архитектура приложения 13](#_Toc73543643)

[4. Описание применения 18](#_Toc73543644)

[Заключение 25](#_Toc73543645)

[Список использованных источников 26](#_Toc73543646)

# Введение

Торговля овощами и фруктами всегда была распространенным и выгодным бизнесом благодаря высокому спросу на продукты. Особое место в ряду продовольственных торговых точек занимают овощные магазины и фруктовые лавки.  Это одна из бизнес идей, которая не сдает позиции при давлении крупных супермаркетов.

Бизнес идея продажи овощей и фруктов не требует наличия специального оборудования и навыков. Спрос на продукцию остается высоким круглый год, независимо от финансовой ситуации в стране. Овощи и фрукты – входят в тройку самых потребляемых продуктов.

Данная курсовая работа называется «Программа магазин для сбыта плодов и овощей». Процесс работы интернет-магазина имеет ряд особенностей. Выделим основные задачи, которые возлагаются на интернет-магазин:

1. Регистрация и учет клиентов
2. Предоставление возможности для выбора и добавления товаров в корзину
3. Возможность оформить заказ

Следовательно, разрабатываемый программный продукт должен отвечать минимальным требованиям, поставленным выше.

# Постановка задачи

## **Описание предметной области**

Рассмотрим магазин, который занимается реализацией фруктов и овощей. Покупатель проходит регистрацию предоставляя некоторые личные данные: ФИО, номер телефона, адрес доставки. После регистрации покупателю предоставляется возможность выбирать продукцию. Покупатель также может просматривать информацию о магазине, редактировать личную информацию.

Предоставляется следующая информация о товаре: цена, справочная информация, наличие, размер скидки. Данная информация загружается из базы данных продуктов.

Покупатель выбирает необходимый товар, который заносится в корзину. Корзину можно редактировать: добавлять товары, удалять, также отображается стоимость будущего заказа.

Когда покупатель определился с выбором, он выбирает вид доставки и оформляет заказ. Кроме того, он может получить специальную скидку и оставить отзыв.

**Задачи проектирования**

Чтобы приступить к проектированию, необходимо выделить минимальный набор основных сущностей и указать типы связи между ними.

В рассматриваемой предметной области можно выделить следующие основные сущности:

При описанной организации данные можно сгруппировать следующим

Клиент – содержит следующую информацию: ФИО, номер телефона, адрес доставки, статус дополнительной скидки и т.д.

Компания – содержит информацию о магазине: название, контактный телефон, адрес, список курьеров, отзывы, время работы.

Курьер – содержит информацию о курьере (ID курьера, ФИО, телефон).

Продукт – содержит следующую информацию: наименование, цена, наличие, справочная информация, скидка.

Скидка – содержит информацию о скидке (наличие, количество).

Доставка – содержит следующую информацию: вид (самовывоз, курьером), стоимость, курьер, адрес доставки, время доставки).

Заказ – содержит следующую информацию: ID заказа, информацию о доставке, информацию о покупателе, список корзин, сумма с доставкой, состояние заказа.

Корзина – содержит информацию о продукте в корзине (наименование, количество, общая сумма).

Состояние заказа – содержит информацию о состоянии заказа (вид, дата оформления).

Администратор – содержит полную информацию о всех базах данных.

Также выделим дополнительные сущности:

Вид продукта.

Вид доставки.

Статус клиента.

Вид состояния заказа.

Типы связей между сущностями.

1:1

Корзина – Скидка.

Продукт – Корзина.

Заказ – Статус заказа.

Заказ – Доставка.

1:n:

Курьер – Доставка.

Заказ – Корзина.

## **Требования к разрабатываемому программному обеспечению**

Готовый программный продукт должен удовлетворять следующим требованиям:

1. Иметь простой и понятный пользователю интерфейс для ввода и редактирования данных;
2. Иметь специальный интерфейс для администрирования
3. Иметь базу данных для пользователей, курьеров, продуктов и т.д. База данных представлена на основе файловой системы;
4. Использовать систему ролей для пользователей:
   1. Администратор (имеет возможность изменять данные о продуктах, курьерах, компании);
   2. Пользователь (имеет возможность изменять только состояние своей корзины)
5. Позволять легко создавать/изменять данные пользователей;
6. Предоставлять актуальную информацию о продуктах (цены, наличие, скидки);
7. Позволять отслеживать информацию о отзывах, новых заказах, прибыли
8. Позволять клиентам выбирать товары и оставлять их в корзине с возможностью последующего оформления заказа

Таким образом, задача данного курсового проекта сводится к разработке приложения для обеспечения реализации овощей с овощебазы. Готовое приложение должно иметь понятный интерфейс и быть удобным в использовании.

# Проектирование задачи

## **Проектирование архитектуры приложения**

Для разработки приложения была написана UML-диаграмма классов, представленная на рис.1. На ней можно видеть сущности и типы связей.

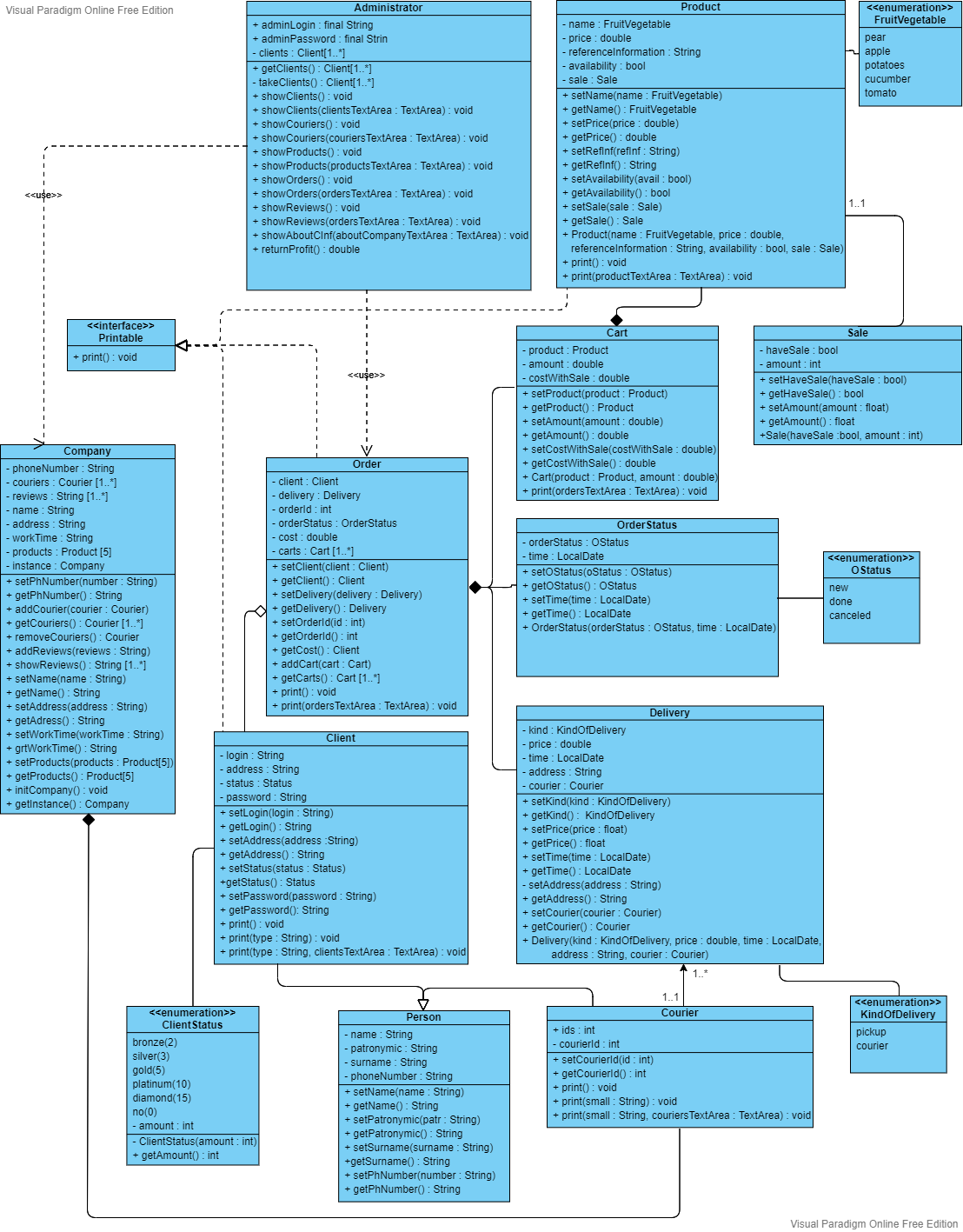


Рис. 1. UML-диаграмма классов

# Используемые технологии

Приложение реализовано на языке программирования Java с использованием технологии JavaFX. В качестве среды разработки была выбрана IntelliJ IDEA 2020. База данных реализована на основе файлов. Для создания дизайна была использована программа SceneBuilder.

## **Среды реализации задачи**

### **IntelliJ IDEA**

IntelliJ IDEA — [интегрированная среда разработки программного обеспечения](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8) для многих языков программирования, в частности [Java](https://ru.wikipedia.org/wiki/Java), [JavaScript](https://ru.wikipedia.org/wiki/JavaScript), [Python](https://ru.wikipedia.org/wiki/Python), разработанная компанией [JetBrains](https://ru.wikipedia.org/wiki/JetBrains).

Первая версия появилась в январе [2001 года](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4) и быстро приобрела популярность как первая среда для Java с широким набором интегрированных инструментов для [рефакторинга](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3), которые позволяли программистам быстро реорганизовывать исходные тексты программ. Дизайн среды ориентирован на продуктивность работы программистов, позволяя сконцентрироваться на функциональных задачах, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций.

Начиная с шестой версии продукта IntelliJ IDEA предоставляет интегрированный инструментарий для разработки графического пользовательского интерфейса. Среди прочих возможностей, среда хорошо совместима со многими популярными [свободными](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) инструментами разработчиков.

IntelliJ IDEA обладает некоторыми наиболее эффективными функциями завершения кода Java. Его алгоритм прогнозирования может точно предполагать, что кодер пытается набрать, и завершает его. IntelliJ IDEA разработан на основе принципа кодирования, согласно которому разработчикам должно быть позволено писать коды с как можно меньшим количеством отвлекающих факторов. Чтобы помочь разработчикам организовать рабочий процесс, IntelliJ IDEA предлагает набор инструментов, который включает в себя декомпилятор, поддержку Docker, средство просмотра байт-кода, FTP и многие другие инструменты.

### **Java**

Java — объектно-ориентированный язык программирования, разрабатываемый компанией Sun Microsystems с 1991 года и официально выпущенный 23 мая 1995 года. Изначально новый язык программирования назывался Oak (James Gosling) и разрабатывался для бытовой электроники, но впоследствии был переименован в Java и стал использоваться для написания апплетов, приложений и серверного программного обеспечения.

Программы на Java могут быть транслированы в байт-код, выполняемый на виртуальной java-машине (JVM) — программе, обрабатывающей байт-код и передающей инструкции оборудованию, как интерпретатор, но значительно быстрее.

Язык Java зародился как часть проекта создания ПО для бытовых приборов. Реализация проекта была начата на языке [C++](http://progopedia.ru/language/c-plus-plus/), но было решено, что необходим платформо-независимый язык программирования, позволяющий создавать программы, которые не приходилось бы компилировать отдельно для каждой архитектуры и можно было бы использовать на различных процессорах под различными операционными системами.

Язык Java потребовался для создания интерактивных продуктов для сети Internet. Java предоставляет синтаксис, сходный с [C](http://progopedia.ru/language/c/) и [C++](http://progopedia.ru/language/c-plus-plus/).

Три ключевых элемента в технологии языка Java:

1. Java предоставляет для широкого использования свои апплеты (applets) — небольшие, надежные, динамичные, не зависящие от платформы активные сетевые приложения.
2. Java сочетает простой и знакомый синтаксис с надежной и удобной в работе средой разработки.
3. Java предоставляет богатый набор классов объектов для ясного абстрагирования многих системных функций, используемых при работе с окнами, сетью и для ввода-вывода. Ключевая черта этих классов заключается в том, что они обеспечивают создание независимых от используемой платформы абстракций для широкого спектра системных интерфейсов.

### **JavaFX**

JavaFX — платформа на основе [Java](https://ru.wikipedia.org/wiki/Java) для создания [приложений](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) с насыщенным [графическим интерфейсом](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F). Может использоваться как для создания настольных приложений, запускаемых непосредственно из-под операционных систем, так и для интернет-приложений ([RIA](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%81%D1%8B%D1%89%D0%B5%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5)), работающих в [браузерах](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D0%B5%D1%80), и для приложений на мобильных устройствах. Платформа JavaFX конкурирует с [Microsoft Silverlight](https://ru.wikipedia.org/wiki/Microsoft_Silverlight), [Adobe Flash](https://ru.wikipedia.org/wiki/Adobe_Flash) и аналогичными системами.

Платформа JavaFX

* включает в себя набор графических и мультимедийных API для создания и развертывания многофункциональных клиентских приложений;
* позволяет быстро создавать многофункциональные кроссплатформенные приложения;
* поддерживает современные графические процессоры с аппаратным ускорением графики;
* позволяет объединять элементы управления графикой, анимацией и пользовательским интерфейсом в одном программном интерфейсе.

### **SceneBuilder**

JavaFX Scene Builder — это инструмент визуального макета, который позволяет пользователям быстро создавать пользовательские интерфейсы приложений JavaFX без программирования. Пользователи могут перетаскивать компоненты пользовательского интерфейса в рабочую область, изменять их свойства, применять таблицы стилей, а код FXML для создаваемого ими макета автоматически создается в фоновом режиме. Результатом является файл FXML, который затем можно объединить с проектом Java, привязав пользовательский интерфейс к логике приложения.

Инструмент визуального макета Scene Builder

* позволяет легко создавать макеты элементов управления пользовательского интерфейса JavaFX, диаграмм, фигур и контейнеров;
* генерирует FXML, язык разметки на основе XML, который позволяет определять пользовательский интерфейс приложения отдельно от логики приложения (также можно открывать и редактировать существующие файлы FXML);
* можно использовать в сочетании с любой IDE Java;
* позволяет предварительно просмотреть, как будет выглядеть пользовательский интерфейс при развертывании;
* написан как приложение JavaFX, поддерживаемое в Windows, Mac OS X и Linux.

# Программная реализация

В данном разделе будет рассмотрена архитектура приложения, а также будут приведены примеры некоторых пользовательских функций и хранимых процедур.

## **Структура приложения**

В разработке программного продукта использована база данных на основе файловой системы. БД представлена файлами для хранения информации о клиентах (ClientsDB), о курьерах (CouriersDB), о заказах (OrdersDB), о продуктах (ProductsDB), а также отзывы (Reviews) и корзины клиентов (в виде папки Carts с файлами; файлы одноименны с логинами, т.к. логины уникальны). В файлах хранятся объекты различных классов в виде строк формата json. Преобразование в json формат проводится с помощью сторонней библиотеки Fastjson. Для безопасной работы с файлами производится обработка исключительных ситуаций.

Пример кода, реализующий обращение к файлу, представлен ниже

*// считывание отзывов из файла в массив*try {  
 Scanner in = new Scanner(new File(Main.*passReviewsFile*), "UTF-8");  
 String check;  
 while (in.hasNextLine()) {  
 check = in.nextLine().trim();  
 this.reviews.add(check + "\n");  
 }  
 in.close();  
}  
catch(FileNotFoundException ex) {  
 System.*out*.println("WRONG IN " + Main.*passReviewsFile* + "FROM Company");  
 System.*out*.println(ex);  
}

Пример строки клиента в формате json, хранящейся в файле ClientsDB:

*{"address":"Minsk","login":"bob123","name":"Bob","password":"123","patronymic":"-","phoneNumber":"+37529888787877","status":"NO","surname":"Smith"}*

## **Архитектура приложения**

Программа разделена на следующие пакеты: classes (хранит пользовательские классы: Product, Client, Courier, Administrator и др.), controllers (хранит классы-контроллеры, которые взаимодействуют с формой – загрузка данных на формы, обработка нажатий на кнопки, переключатели и др.), enums (отдельный класс для перечислений), fxml (хранит fxml файлы с кодом для графического представления: product.fxml, client.fxml, administrator.fxml и др.). Каждому файлу формата fxml соответствует класс-контроллер для обработки событий на форме.

Ниже представлен пример кода, который обрабатывает событие нажатия на кнопку «В корзину» - реализует закрытие окна каталога и переход в окно корзины:

*// кнопка Корзина*cartButton.setOnAction(event -> {  
  
 *// спрятать старое окно* cartButton.getScene().getWindow().hide();  
 *// найти новое окно* FXMLLoader loader = new FXMLLoader();  
 loader.setLocation(getClass().getResource("/sample/fxml/cart.fxml"));  
 *// загрузить его* try {  
 loader.load();  
 }  
 catch (IOException e){

System.*out*.print("WRONG IN cart.fxml from CatalogController");  
 e.printStackTrace();  
 }  
 *// отобразить новое окно* Parent root = loader.getRoot();  
 Stage newWindow = new Stage();  
 newWindow.setScene(new Scene(root));  
 *// невозможно менять размеры окна* newWindow.setResizable(false);  
  
 newWindow.getIcons().add(new Image("file:C:\\YCHEBA\\4сем\\ООП\\Project\\src\\sample\\images\\onion.png"));  
 newWindow.show();  
});

В классах контроллерах предусмотрена обработка особых ситуаций, когда необходимо запретить пользователю некоторые действия. Например, запрет добавления выбора продукта в корзину при его отсутствии, отсутствие возможности выбора флажка доставки курьером при отсутствии курьеров в базе данных и др.

Ниже приведен пример кода, запрещающий пользователю выбора доставки курьером, если общая цена заказа менее 20 р. В этом случае автоматически «выбирается» флажок доставки самовывозом, иначе (если беспрепятственно выбрана доставка курьером) на форме заполняются поля данных о курьере, стоимость и т.д.

*// нажатие на кнопку Курьер (запрет, если цена < 20 р.)*private void courierClicked() {  
 *// доставка от 20 р.* if (wholeCost < 20 || courier == null) {  
 yourselfRadioButton.fire();  
 }  
 else {  
 deliveryCostLabel.setText(String.*format*("%.2f p.", deliveryPrice));  
 courierLabel.setText(courier.getName() + " " + courier.getSurname() + " ("  
 + courier.getPhoneNumber() + ")");  
 wholeCostLabel.setText(String.*format*("%.2f p.", deliveryPrice + wholeCost));  
 }  
}

При входе в приложение проводится поиск на соответствие введенного логина и пароля данным, содержащимся в базе данных клиентов. Для этого используется итератор по файлу, который открывает файл, считывает строку с обработкой возможных ошибок и исключений, и возвращает следующего клиента из файла.

Перегруженный метод hasNext(), реализованный в классе итератора:

@Override  
public Object next() {  
 Object res;  
 try {  
 String result = nextline;  
 if (nextline != null) {  
 nextline = in.readLine();  
 if (nextline == null) {  
 in.close();  
 }  
 }  
 res = sample.classes.MyJSON.*fromJSON*(result, typeParameterClass);  
 } catch (IOException e) {  
 throw new IllegalArgumentException(e);  
 }  
 return res;  
}

Приведем также пример функции для авторизации пользователя, использующей итератор:

private void loginUser(String loginText, String loginPassword) {  
  
 if (loginText.equals(Administrator.*adminLogin*) && loginPassword.equals(Administrator.*adminPassword*)){  
 openAdminForm();  
 }  
 else {  
 Client client = null;TextFileReader<Client> textFileReader = new TextFileReader<>(Main.*passClientsDB*, Client.class);  
 for (MyIterator iter = textFileReader.getIterator(); iter.hasNext();) {  
 client = (Client) iter.next();  
 if (client.getLogin().equals(loginText) && client.getPassword().equals(loginPassword)) {  
 setCurrentClient(client);  
 return;  
 }  
 }  
 *// вызов формы с сообщением о неверном вводе с возвратом к исходной форме* FaultForm(false);  
 login\_field.setText("");  
 password\_field.setText("");  
 }  
}

Для реализации класса Company был использован паттерн ООП Singleton (одиночка). Паттерн Singleton гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа. Singleton в классе был реализован с синхронизацией:

public static synchronized Company getInstance() {  
 if (*instance* == null) {  
 *instance* = new Company();  
 }  
 return *instance*;  
}

Для подсчета общей прибыли магазина в классе Administrator реализована функция, которая считывает заказы из файла и суммирует их стоимость, а также производит округление до денежных единиц (2 знака после точки) встроенным методом round:

public double returnProfit(){  
  
 double profit = 0;  
  
 *// считывание заказов из файла* try {  
 Scanner in = new Scanner(new File(Main.*passOrdersDB*), "UTF-8");  
 String s;  
 while (in.hasNextLine()) {  
 s = in.nextLine();  
 *// обработка на пустоту файла* try {  
 profit += JSON.*parseObject*(s, Order.class).getCost();  
 }catch (Exception ex) {  
 s = null;  
 }  
 }  
 in.close();  
 }  
 catch(FileNotFoundException ex) {  
 System.*out*.println("WRONG IN OrdersDB.txt FROM Administrator (showProfit)");  
 System.*out*.println(ex);  
 }  
 *// округление до сотых* profit = (int)Math.*round*(profit\*100) / 100.0;  
  
 return profit;  
}

# Описание применения

Приложение содержит следующие классы-контроллеры с соответствующими формами: Authorize Controller, Administrator Controller, Sign Up Controller, Catalog Controller, About Company Controller, Cart Controller, Order Controller, Contacts Controller, Personal Cabinet Controller, Apple (/Pear/Potato/Tomato/Cucumber) Controller, контроллеры-подтверждения.

Опишем работу с каждым из них:

* Authorize Controller

Чтобы войти в приложение необходимо ввести пользовательский логин и пароль или зарегистрироваться, нажав на соответствующую кнопку (рис. 2).

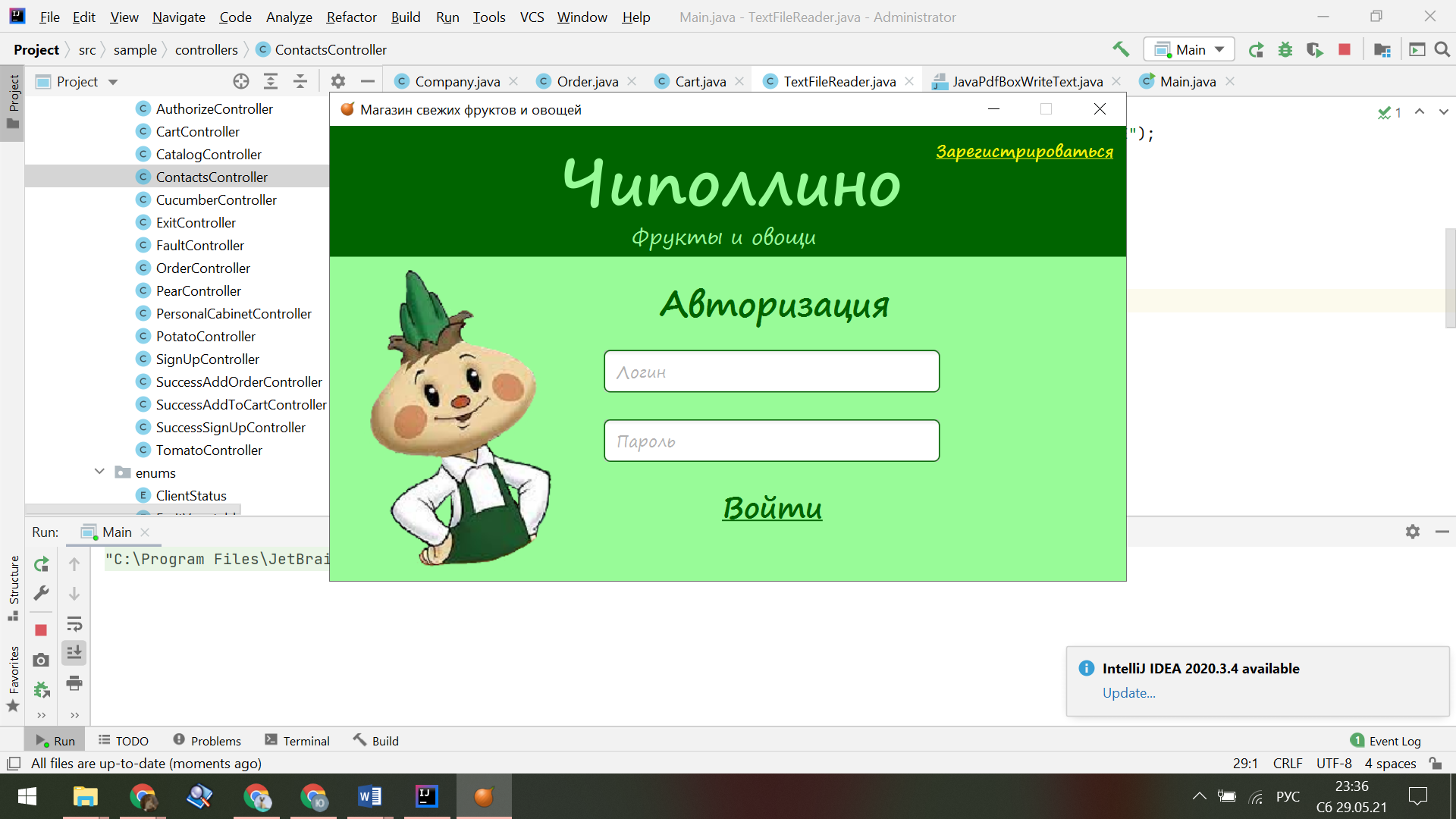


Рис. 2. Форма авторизации.

* Administrator Controller

При входе от администратора (логин: administrator, пароль: rotartsinimda), появляется окно администрирования, предоставляющее следующие возможности: просмотр, поиск, изменение статуса клиентов; просмотр, поиск, удаление, добавление курьеров; просмотр, изменение продуктов; просмотр, поиск заказов, подсчет прибыли; просмотр, удаление отзывов; просмотр, изменение данных компании (телефон, адрес, время работы) (рис. 3).

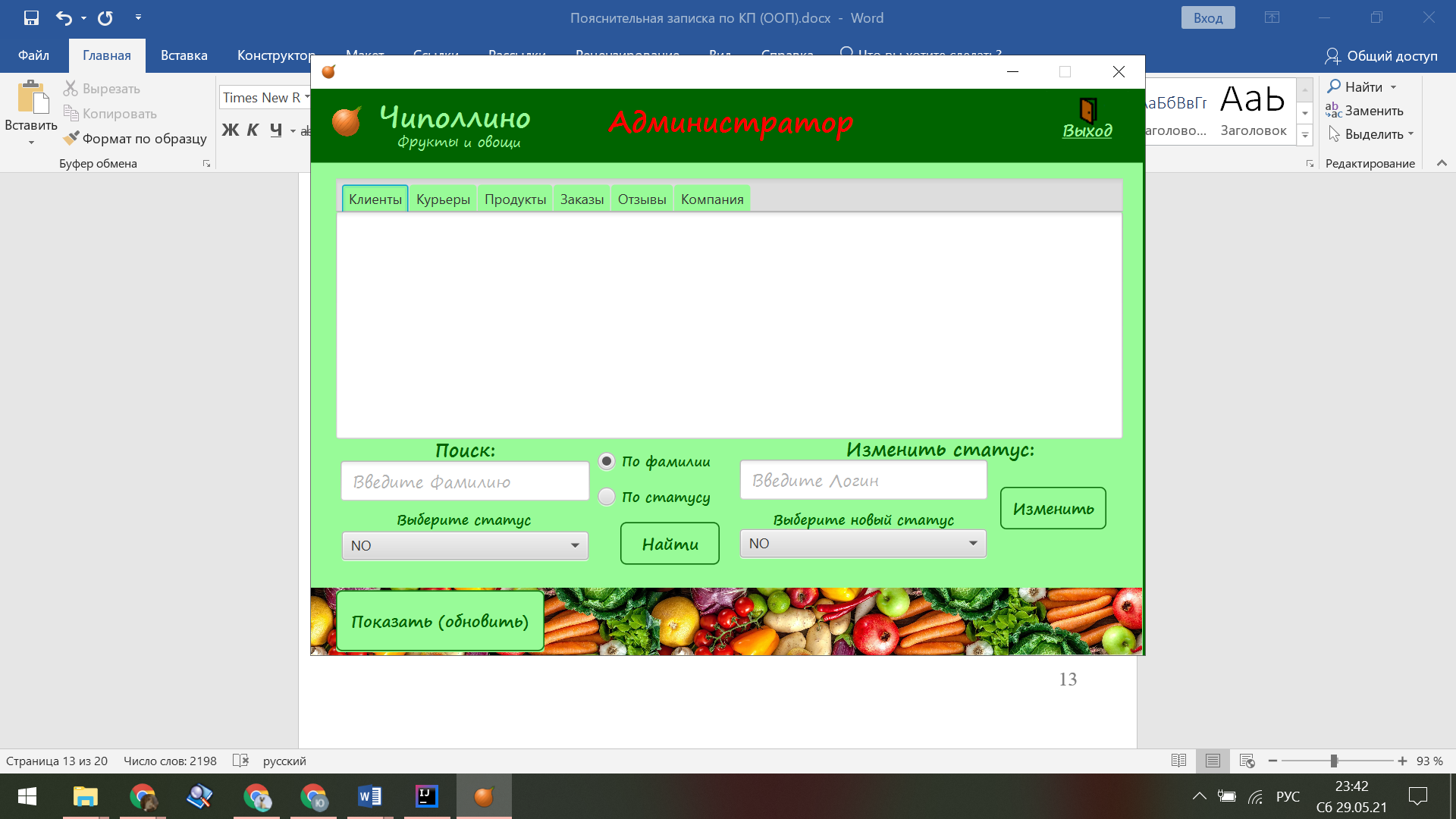


Рис. 3. Форма администрирования.

* Sign Up Controller

Чтобы зарегистрироваться, пользователь должен заполнить все обязательные поля ввода. И подтвердить регистрацию. После ввода необходимых данных и регистрации информация будет сохранена в системе (добавлена новая строка в файл базы данных клиентов), о чем сообщит окно успешной регистрации, в противном случае появится окно ошибки. Также предусмотрена валидация полей, чтобы не допустить ввод некорректных данных (рис. 4).

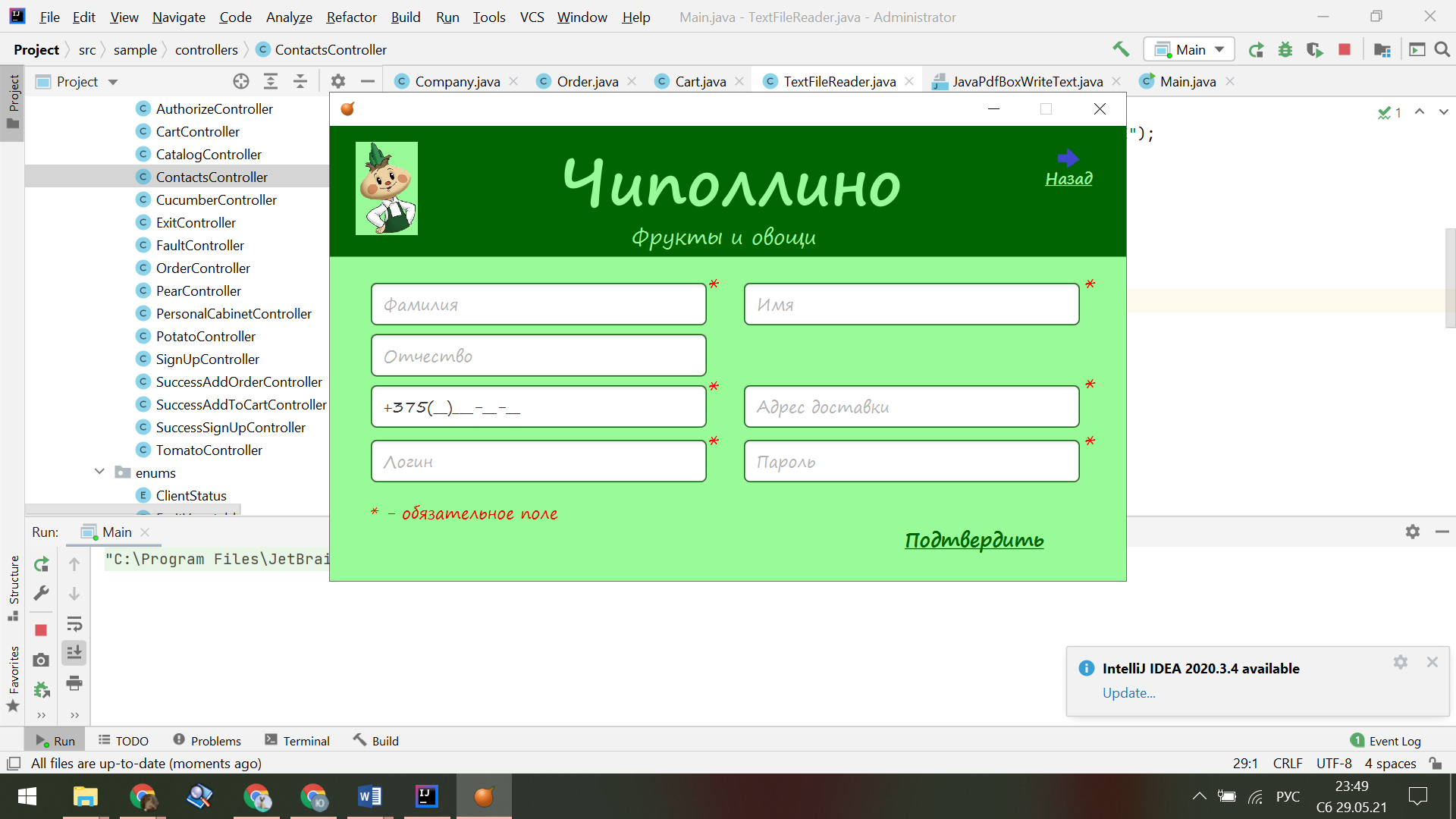


Рис. 4. Форма регистрации.

* Catalog Controller

Окно каталога предлагает на выбор продукты (яблоко, огурец, картофель, груша, томат), а также возможность перехода в корзину. Сверху формы представлено меню кнопок для просмотра контактной информации, личного данных пользователя (рис. 5).

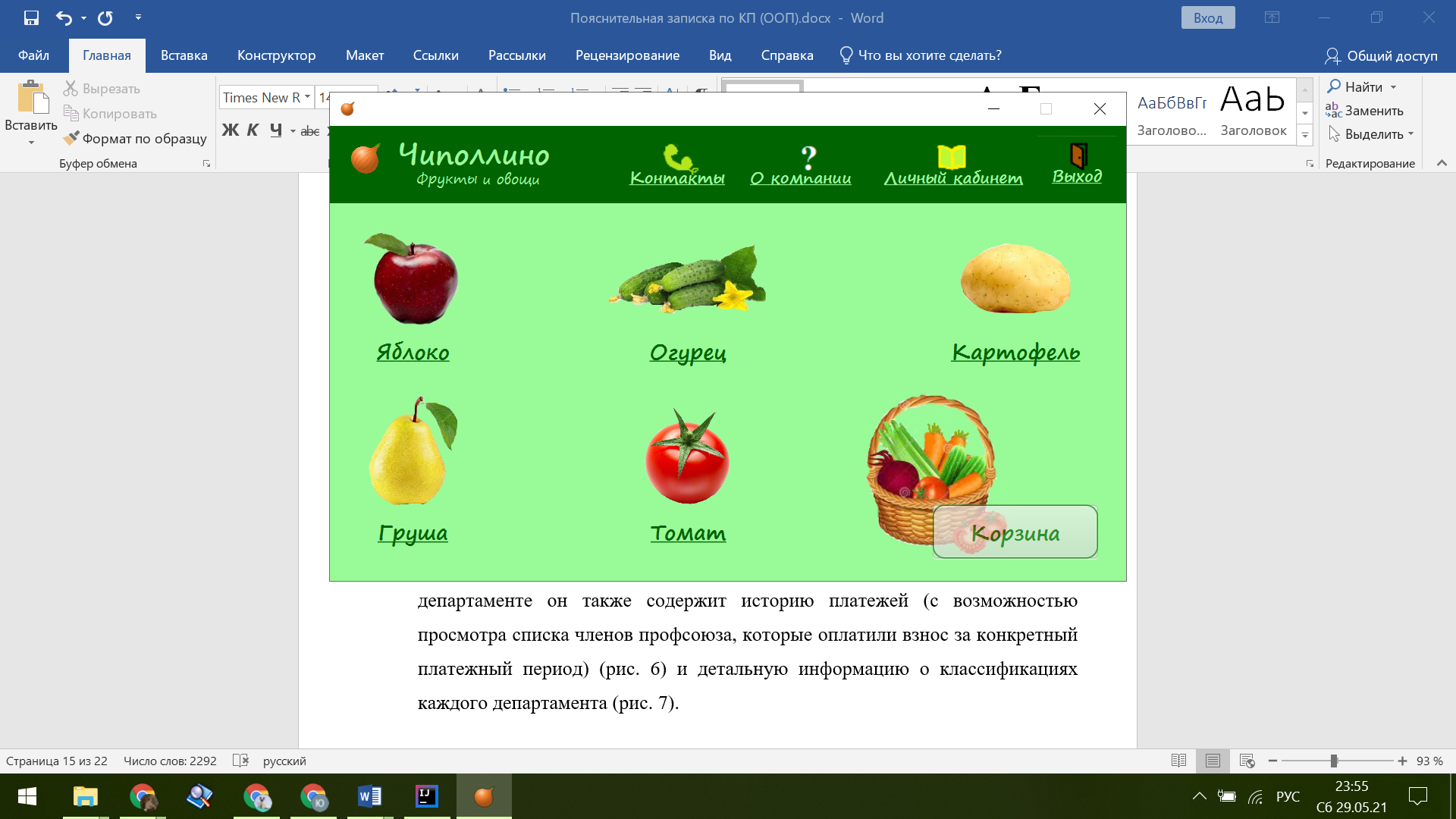


Рис. 5. Форма каталога.

* About Company Controller

В окно, представляющее информацию о компании, можно перейти, нажав кнопку «О компании» в окне каталога. В данном окне можно прочесть чем занимается магазин, узнать цели и др. (рис. 6).

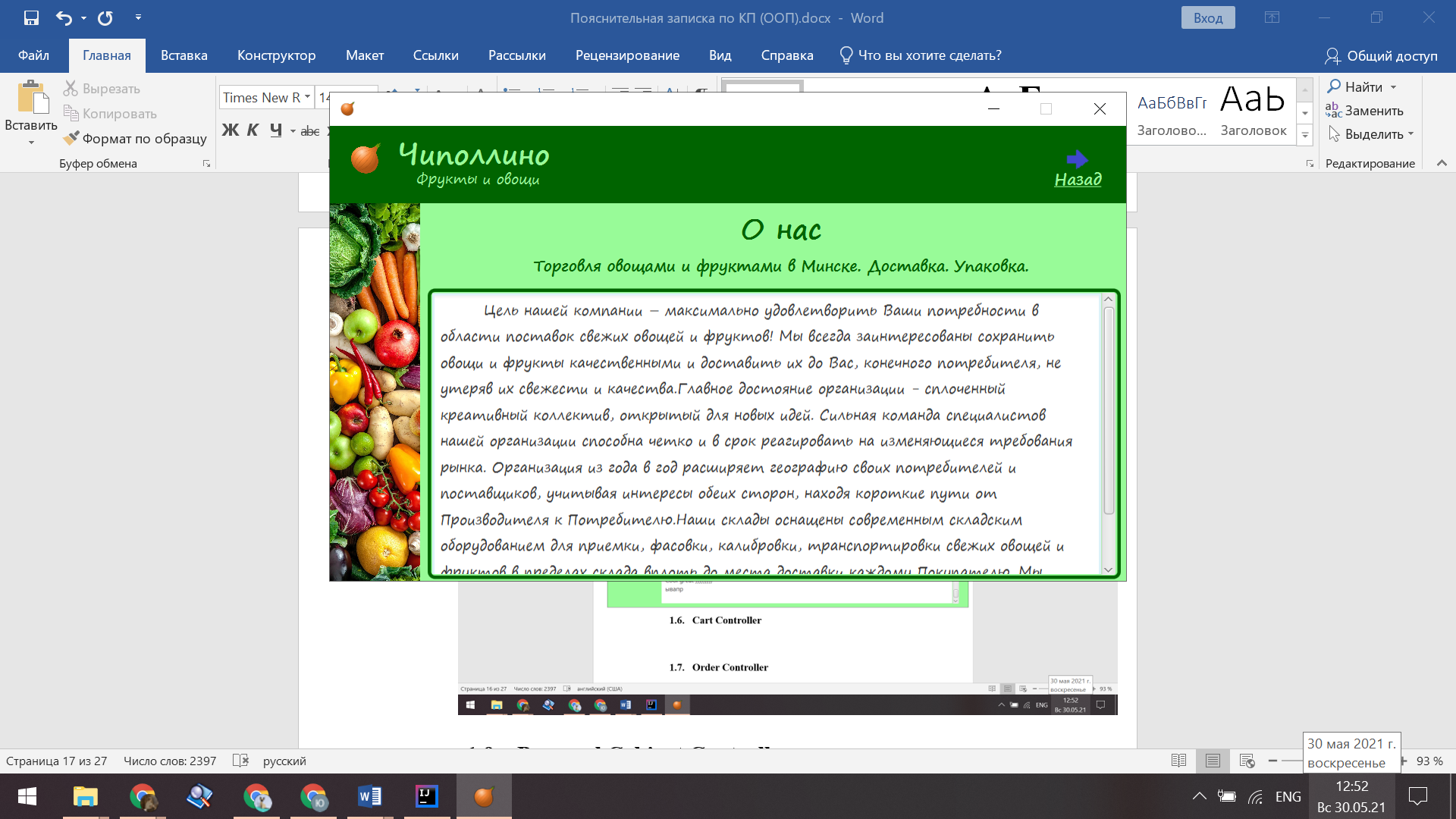


Рис. 6. Форма с информацией о компании.

* Cart Controller

Окно корзина показывает содержимое корзины покупателя в данный момент. Любой продукт можно удалить из корзины, нажав на соответствующую кнопку, также указывается общая сумма продуктов в корзине (рис. 7).

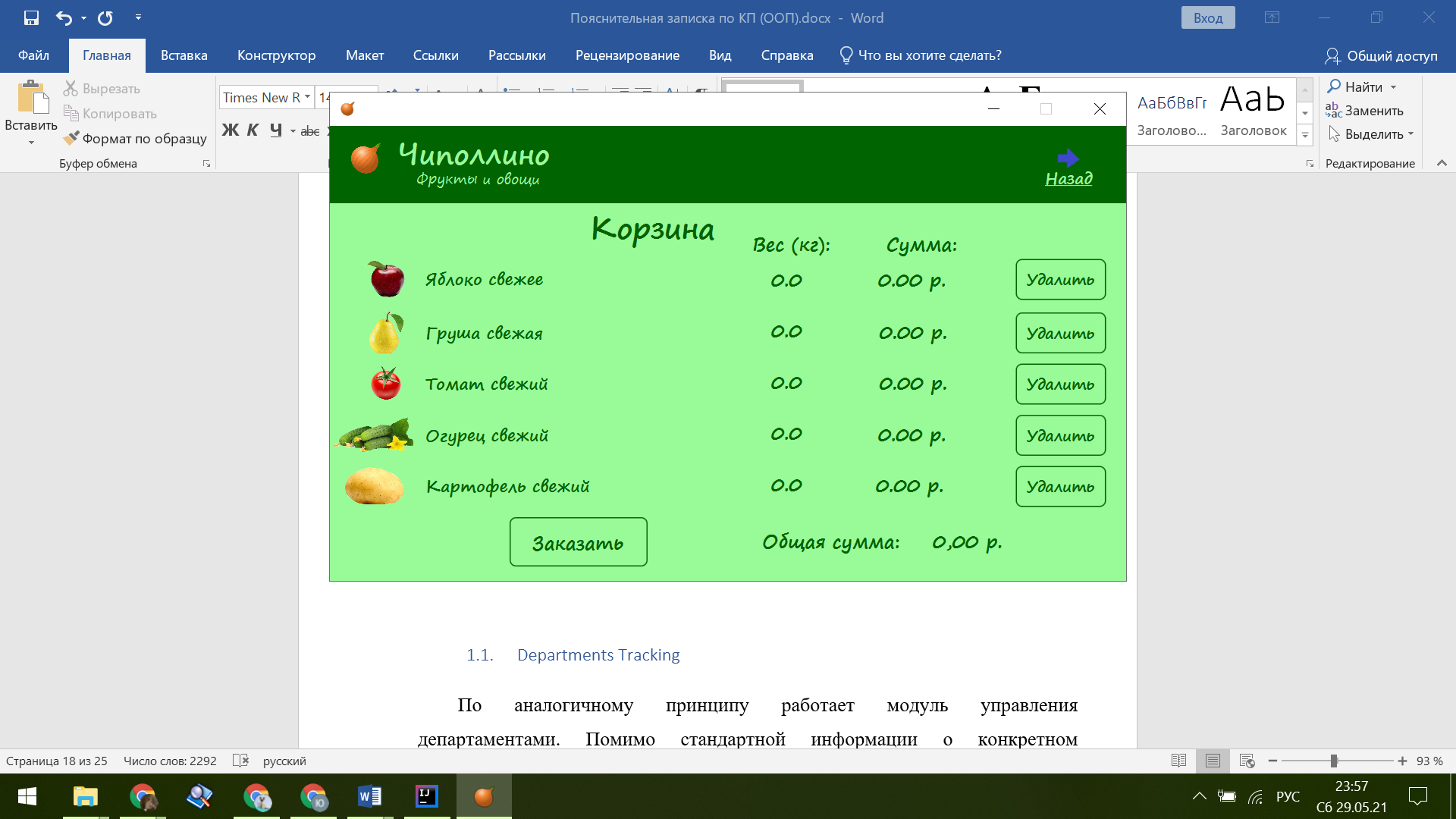


Рис. 7. Форма корзины.

* Order Controller

Окно заказа показывает некоторую информацию о заказе (номер заказа, дополнительную скидку клиента, вид доставки, сумму к оплате с учетом скидки). Покупатель может выбрать вид доставки и подтвердить/отменить заказ (рис. 8).

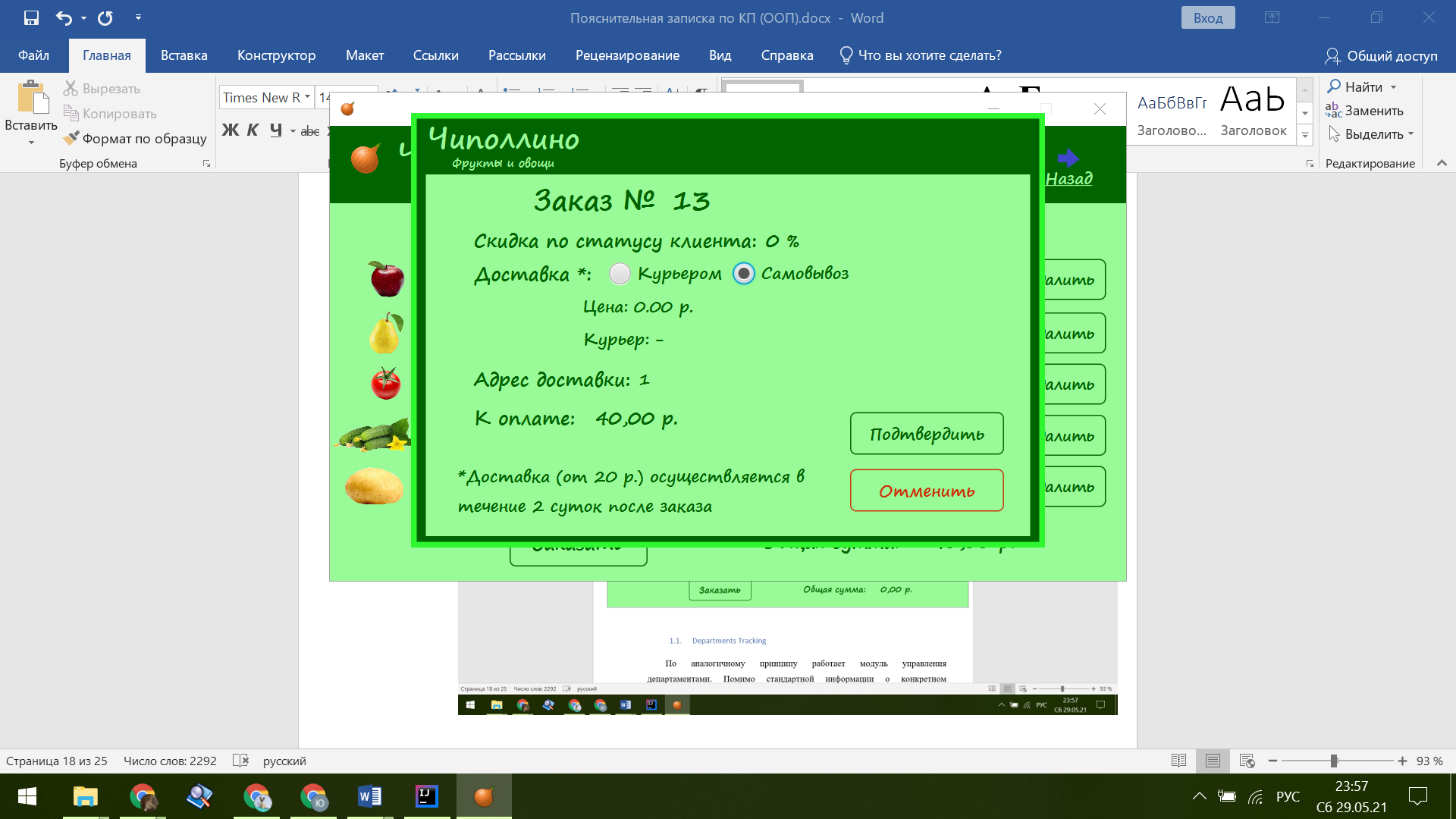


Рис. 8. Форма заказа.

* Contacts Controller

В окно с контактной информацией магазина можно перейти, нажав соответствующую кнопку в окне каталога. Также в этом окне можно оставлять отзывы и пожелания (рис. 9).

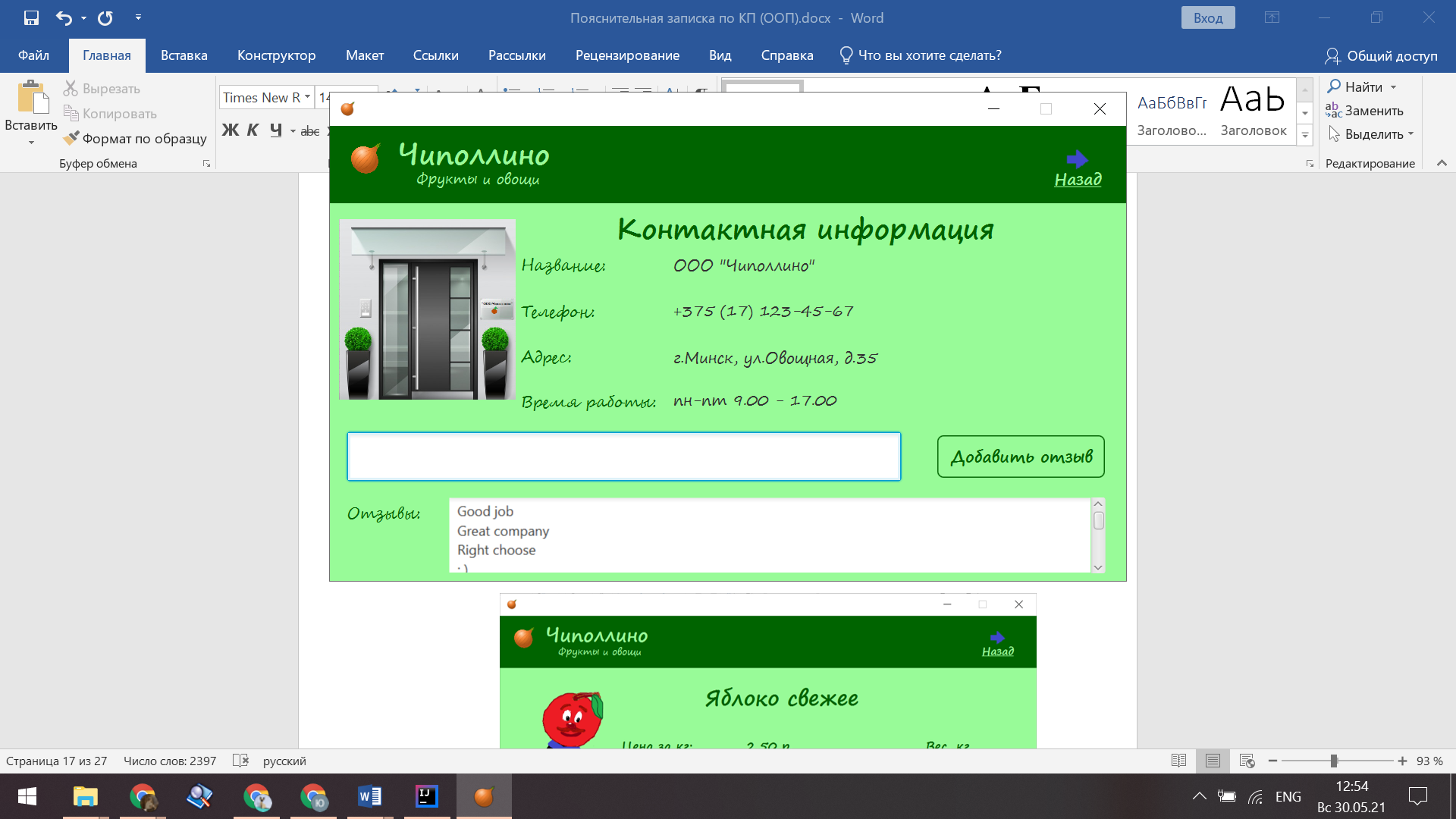


Рис. 9. Форма с контактами и отзывами.

* Personal Cabinet Controller

Окно личный кабинет позволяет просмотреть пользователю введенную при регистрации информацию и при необходимости изменить ее (логин пользователя изменению не подлежит) (рис. 10).

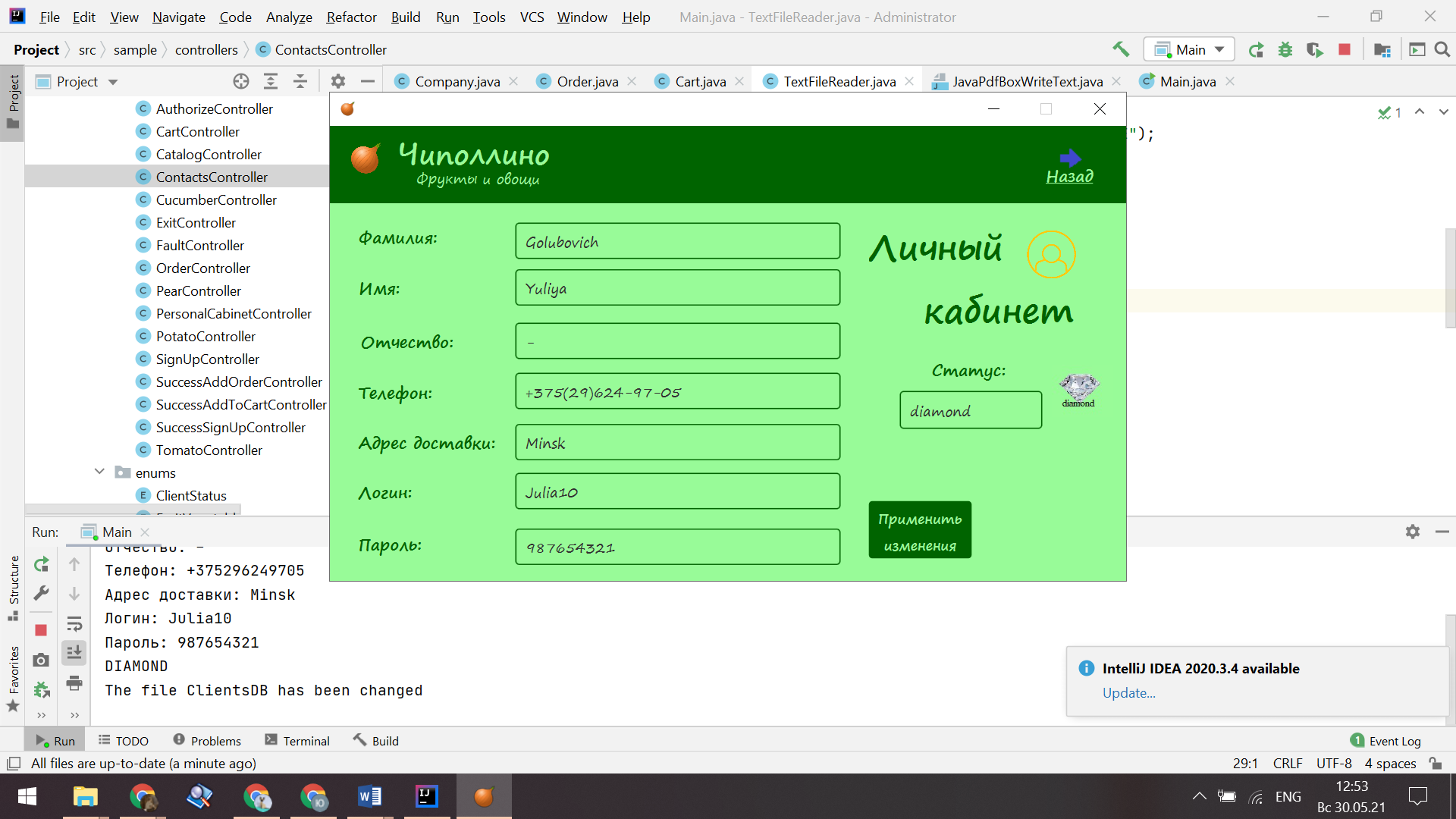


Рис. 10. Форма личного кабинета.

* Apple (/Pear/Potato/Tomato/Cucumber) Controller

Данные окна представляют информацию о конкретном продукте: цена, наличие на складе, размер скидки. Перейти к окну продукта можно нажав на кнопку с названием продукта в окне каталога. Покупатель может выбрать необходимое количество продукта и добавить его в корзину, нажав на кнопку «В корзину» (рис. 11).

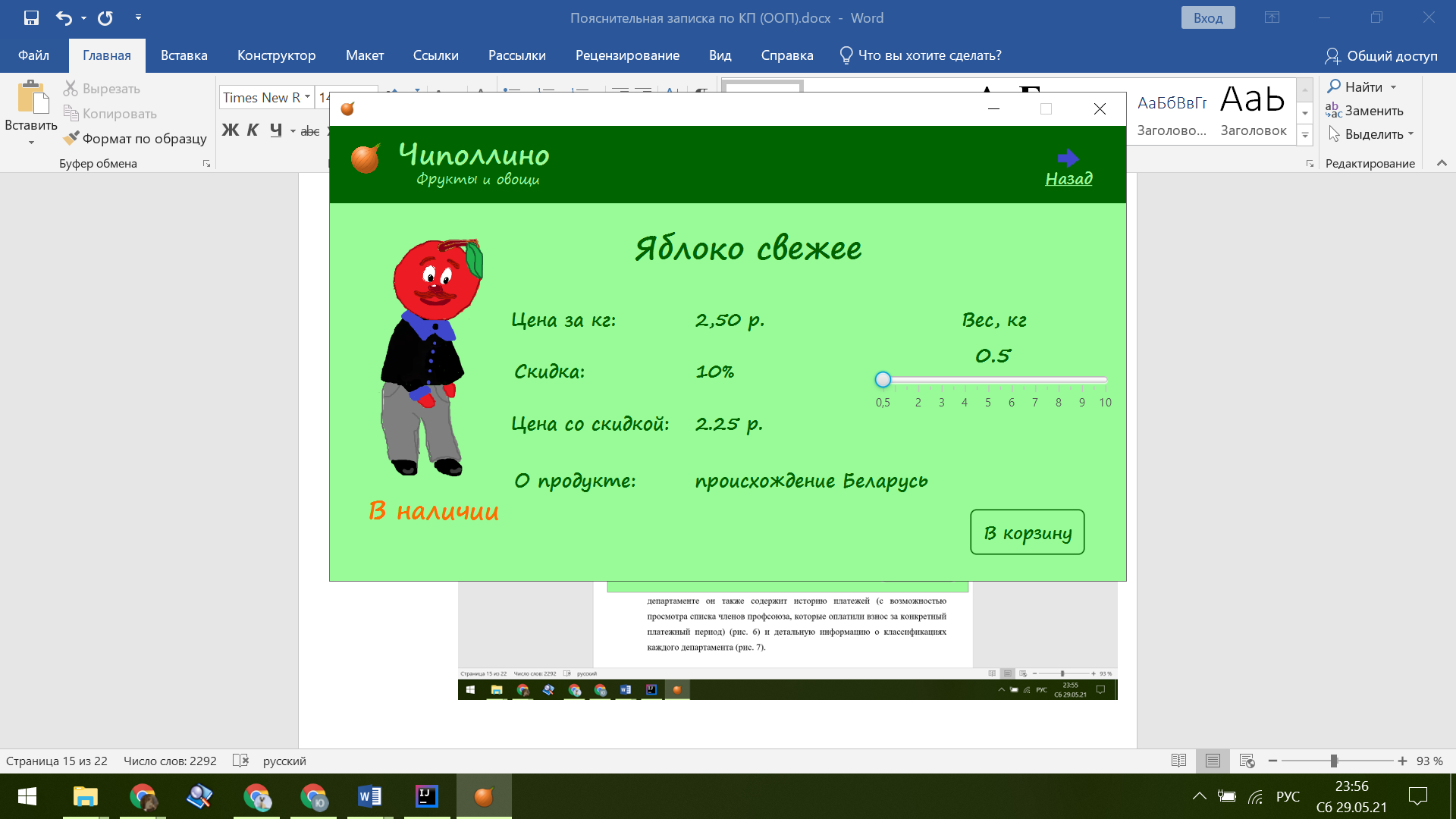
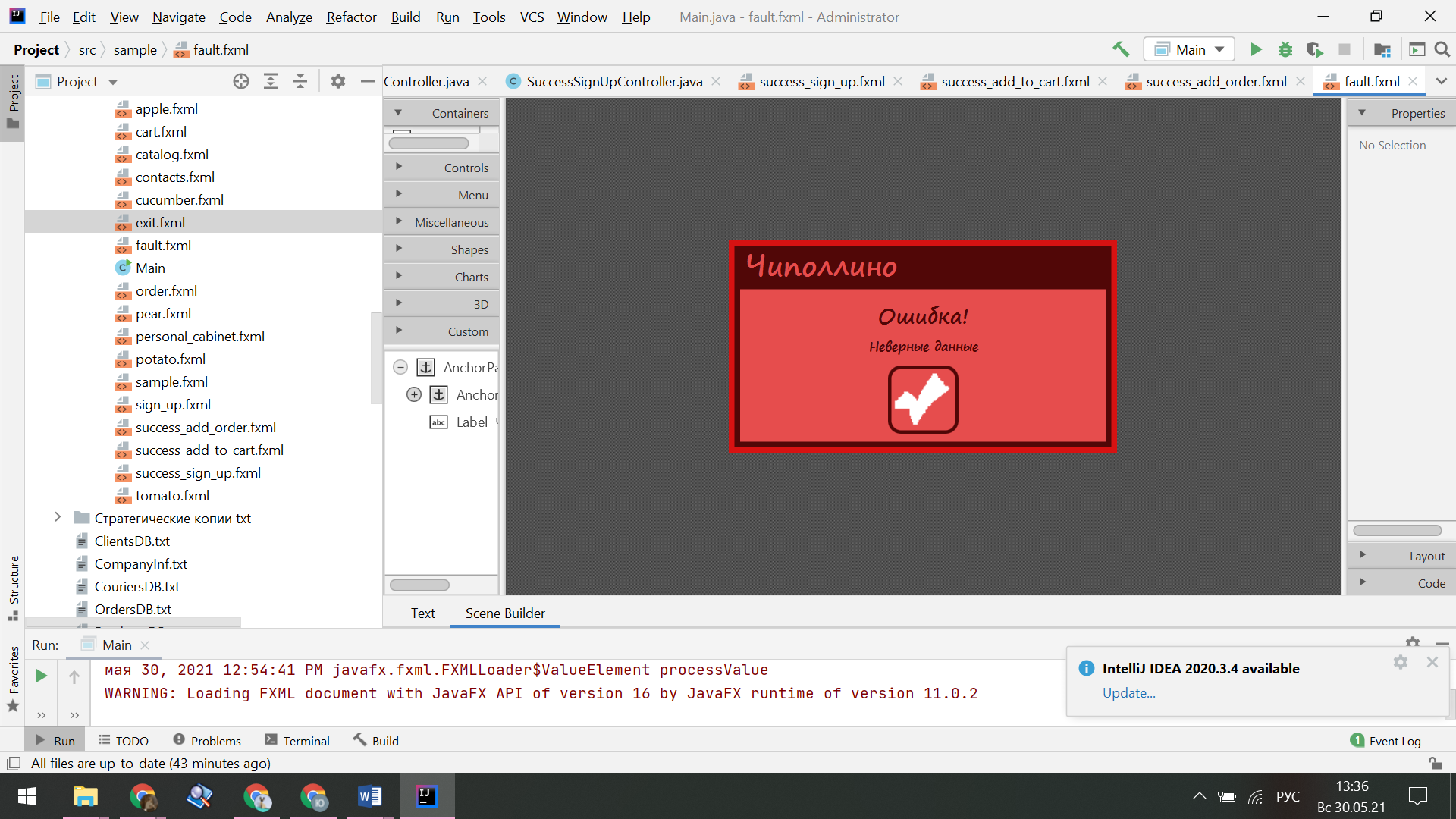
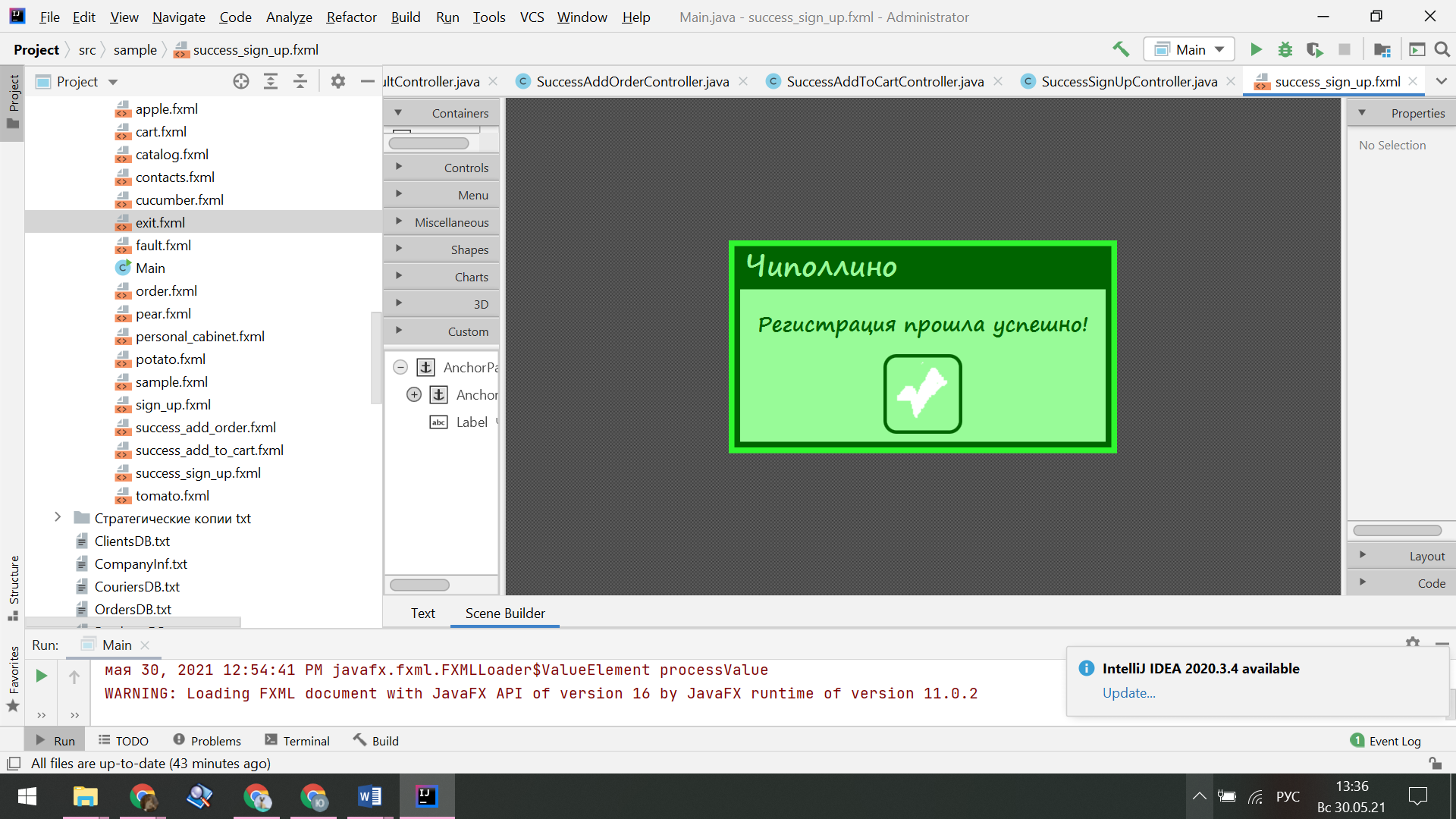
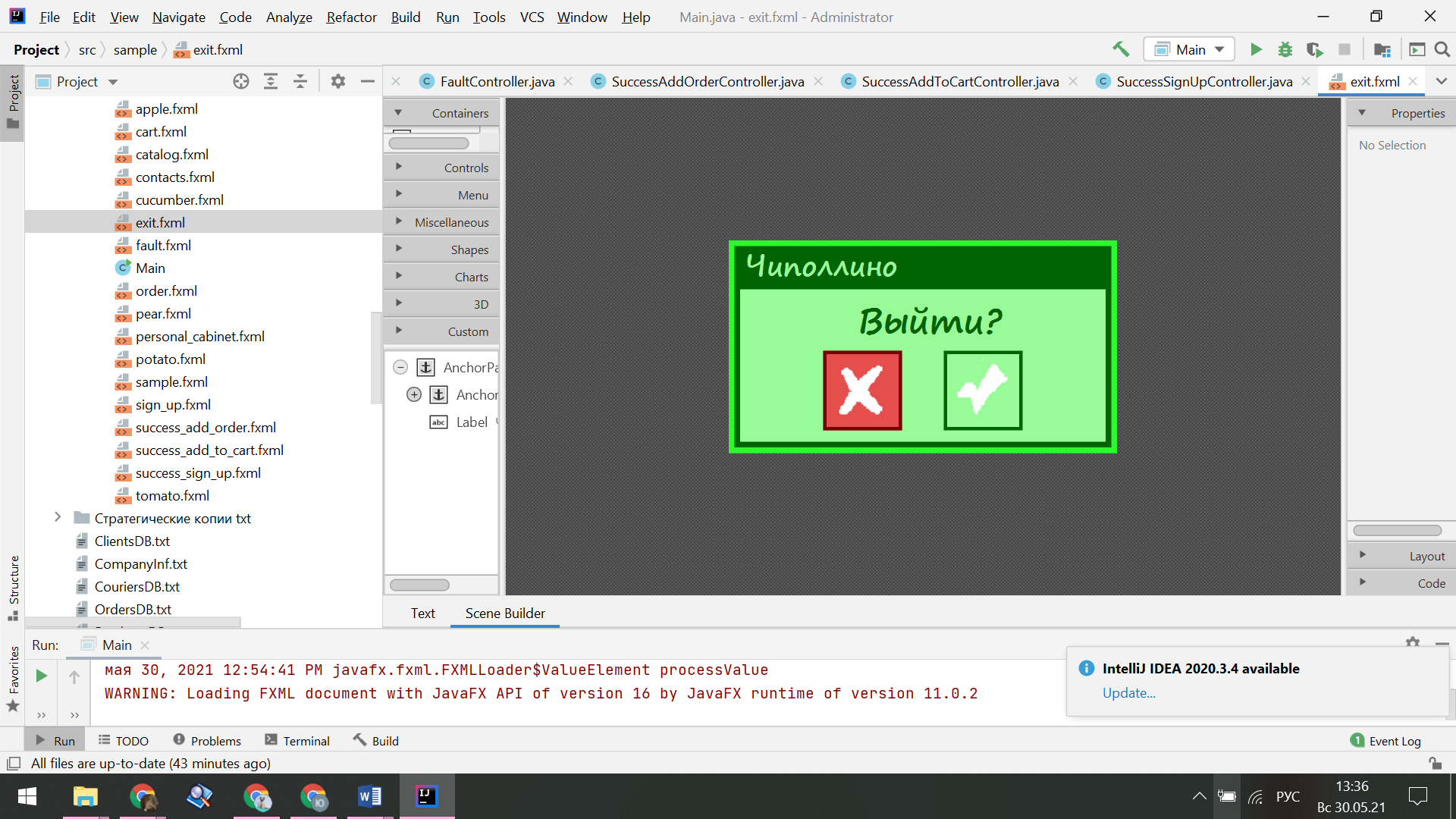


Рис. 11. Форма яблока.

* Контроллеры-подтверждения

Котроллеры-подтверждения представлены классами Fault Controller, Exit Controller, Success Add Order Controller, Success Add To Cart Controller, Success Sign Up Controller. Это дополнительные окна, представляющие сообщения об ошибке или успешном завершении действия, они позволяют лишь подтвердить закрытие (рис. 12).

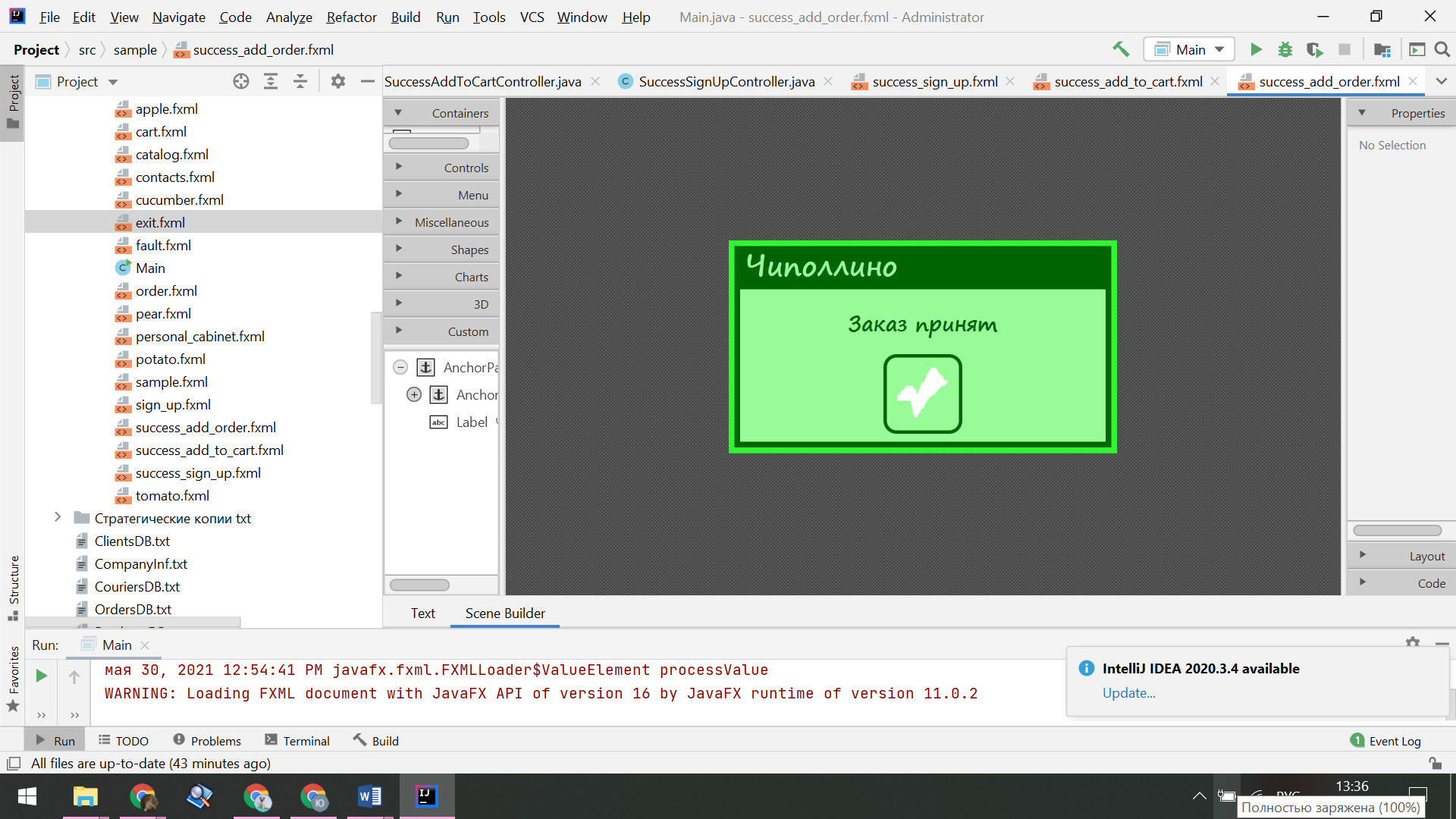
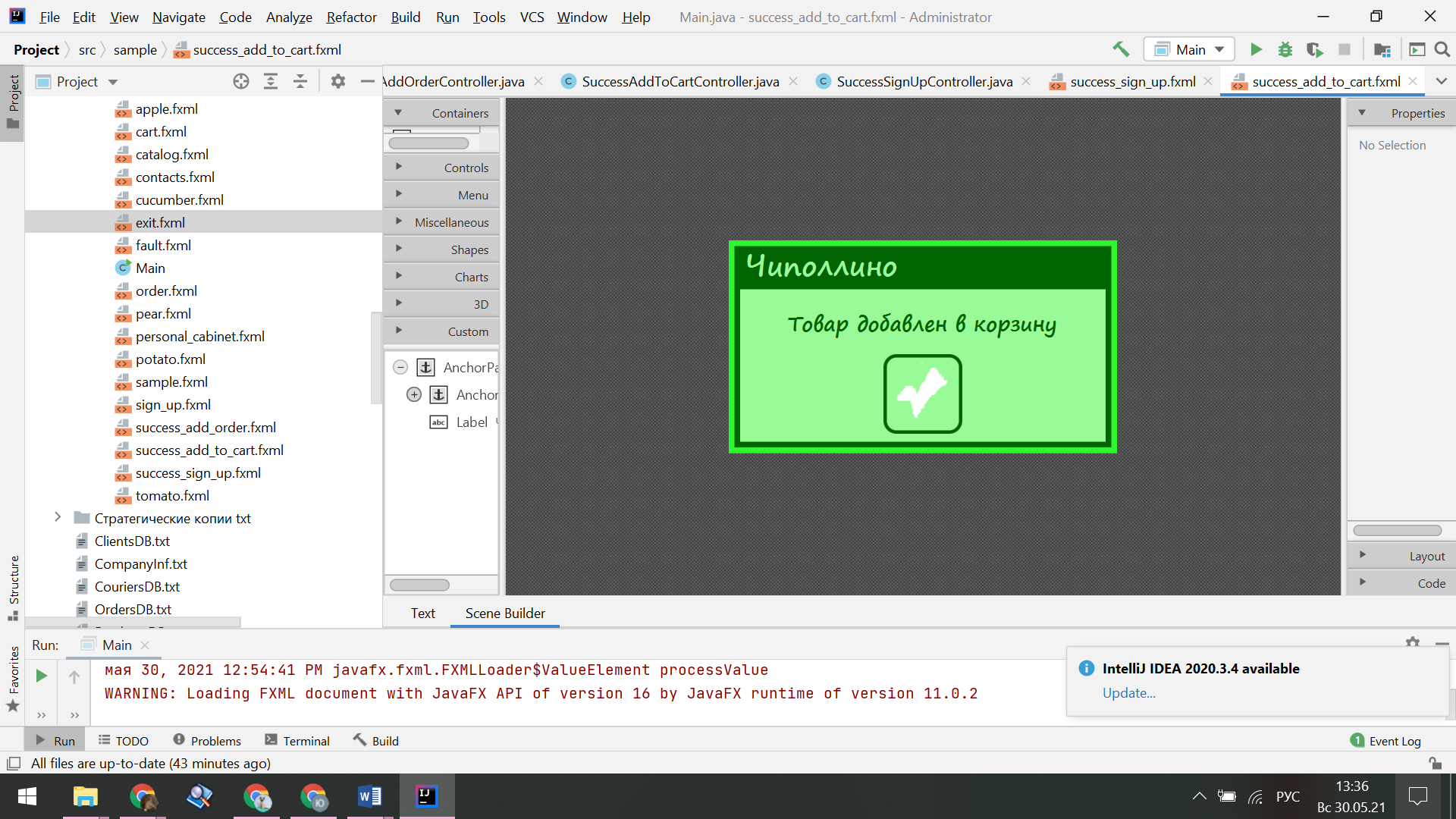
 

Рис. 12. Формы-подтверждения.

# Заключение

В рамках работы над курсовым проектом был разработан программный проект для реализации продуктов (продажи овощей и фруктов).

*Оценивая проделанную работу, можно сделать выводы:*

 Приложение предоставляет пользователям наиболее необходимые услуги, а именно возможность составить корзину товаров, сделать заказ и выразить пожелания администрации (оставить отзывы).

 Дизайн приложения интуитивно понятный, удобный и простой.

В приложении реализована навигация с помощью кнопок. Кнопки переходов легко отличить от основного текста, т.к. они расположены в одном месте.

 Особое внимание уделяется безопасности пользователей и контролю корректности вводимых данных.

Реализован интерфейс администратора, позволяющий плохо подготовленным работникам делать меньше ошибок при работе с файловой базой данных.

За время написания проекта была изучена информация о работе интернет-магазинов.

Для написания проекта были изучены основы языка программирования Java, а также технология JavaFX. В программе использовался язык разметки FXML. Разработка производилась в среде IntelliJ IDEAс использованием приложения SceneBuilder для работы с дизайном.

Были проанализированы различные подходы и технологии, из которых были выбраны наиболее подходящие.

Приложение можно доработать: расширить функциональность приложения, изменить дизайн и др.

# Список использованных источников

1. Шилдт Г. Java 8. Полное руководство - М.: ООО «И.Д. Вильямс», 2015. - 1377 с.
2. [МакГрат М.](https://ru.pdfdrive.com/search?q=%D0%9C%D0%B0%D0%BA%D0%93%D1%80%D0%B0%D1%82+%D0%9C.) Программирование на Java для начинающих – М.: Издательство «Э», 2016. - 192 с.
3. Руководство по [JavaFX 8 - Oracle Help Center: https://docs.oracle.com/javase/8/javafx/api/javafx/](JavaFX%208%20-%20Oracle%20Help%20Center:%20%20%20%20%20https://docs.oracle.com/javase/8/javafx/api/javafx/)
4. Руководство по языку программирования Java: <https://metanit.com/java/tutorial/>
5. Руководство по JavaFX: https://metanit.com/java/javafx/
6. <https://refactoring.guru/ru/design-patterns> - паттерны проектирования
7. <https://habr.com/ru/post/129494/> - Singleton в Java
8. <https://habr.com/ru/post/136766/> - Паттерны ООП в метафорах
9. <https://github.com/vas7n/VAMaskField> - библиотека маски ввода
10. <https://betacode.net/11009/javafx> - Руководства JavaFX
11. <https://spec-zone.ru/RU/Java/FX/8/docs/api/javafx/scene/doc-files/cssref.html> - JavaFX Справочник
12. <https://github.com/alibaba/fastjson/> - библиотека для формата json
13. <http://progopedia.ru/language/java/> - Язык программирования java
14. <https://ru.wikipedia.org/wiki/IntelliJ_IDEA> - IntelliJ IDEA
15. <https://russianblogs.com/article/136832074/> - технология JavaFX
16. <https://www.oracle.com/java/technologies/javase/javafxscenebuilder-info.html> - конструктор сцен SceneBuilder