Введение

В рамках курсового проекта по предмету «Объектно-ориентированное программирование» мне была поставлена задача реализовать библиотеку классов, разработать консольное приложение для чтения и записи из файлов по теме «Оптовая база».

Для хранения информации о товарах, поставщиках и поставках были использованы документы Microsoft Excel а также текстовые файлы.

Microsoft Excel – это программа, предназначенная для составления электронных таблиц, оформления финансовых документов, программирования сложных расчётов и построения графиков на базе числовых вычислений.

1. Теоретические основы
   1. Обзор возможностей языка C# и платформы .Net

На сегодняшний момент язык программирования C# один из самых мощных, быстро развивающихся и востребованных языков в ИТ-отрасли. В настоящий момент на нем пишутся самые различные приложения: от небольших десктопных программок до крупных веб-порталов и веб-сервисов, обслуживающих ежедневно миллионы пользователей.

По сравнению с другими языками C# достаточно молодой, но в то же время он уже прошел большой путь. Первая версия языка вышла вместе с релизом Microsoft Visual Studio .NET в феврале 2002 года. Текущей версией языка является версия C# 8.0, которая вышла в сентябре 2019 года вместе с релизом .NET Core 3.

C# является объектно-ориентированным и в этом плане много перенял у Java и С++. Например, C# поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. И C# продолжает активно развиваться, и с каждой новой версией появляется все больше интересных функциональностей, как, например, лямбды, динамическое связывание, асинхронные методы и т.д.

Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие её основные черты:

Поддержка нескольких языков. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) - своего рода ассемблер платформы .NET. Поэтому мы можем сделать отдельные модули одного приложения на отдельных языках.

Кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент .NET Core поддерживается на большинстве современных ОС Windows, MacOS, Linux. Используя различные технологии на платформе .NET, можно разрабатывать приложения на языке C# для самых разных платформ - Windows, MacOS, Linux, Android, iOS, Tizen.

Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на C# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET.

Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET и Entity Framework Core. Для построения графических приложений с богатым насыщенным интерфейсом - технология WPF и UWP, для создания более простых графических приложений - Windows Forms. Для разработки мобильных приложений - Xamarin. Для создания веб-сайтов - ASP.NET и т.д.

* 1. Обзор принципов гибкой разработки ПО

Гибкая методология разработки (англ. Agile software development), agile-методы — обобщающий термин для целого ряда подходов и практик, основанных на ценностях Манифеста гибкой разработки программного обеспечения и 12 принципах, лежащих в его основе.

К гибким методологиям, в частности, относят экстремальное программирование, DSDM, Scrum, FDD, BDD и др.

Agile — семейство процессов разработки, а не единственный подход в разработке программного обеспечения, и определяется Agile Manifesto. Agile не включает практик, а определяет ценности и принципы, которыми руководствуются команды.

Основные идеи:

* Люди и взаимодействие важнее процессов и инструментов;
* Работающий продукт важнее исчерпывающей документации;
* Сотрудничество с заказчиком важнее согласования условий контракта;
* Готовность к изменениям важнее следования первоначальному плану.

Принципы, которые разъясняет Agile Manifesto:

* Удовлетворение клиента за счёт ранней и бесперебойной поставки ценного программного обеспечения;
* Приветствие изменений требований даже в конце разработки (это может повысить конкурентоспособность полученного продукта);
* Частая поставка рабочего программного обеспечения (каждый месяц или неделю или ещё чаще);
* Тесное, ежедневное общение заказчика с разработчиками на протяжении всего проекта;
* Проектом занимаются мотивированные личности, которые обеспечены нужными условиями работы, поддержкой и доверием;
* Рекомендуемый метод передачи информации — личный разговор (лицом к лицу);
* Работающее программное обеспечение — лучший измеритель прогресса;
* Спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный темп на неопределённый срок;
* Постоянное внимание улучшению технического мастерства и удобному дизайну;
* Простота — искусство не делать лишней работы;
* Лучшие технические требования, дизайн и архитектура получаются у самоорганизованной команды;
* Постоянная адаптация к изменяющимся обстоятельствам. Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

Существуют методологии, которые придерживаются ценностей и принципов заявленных в Agile Manifesto, некоторые из них:

Agile Modeling (англ.) — набор понятий, принципов и приёмов (практик), позволяющих быстро и просто выполнять моделирование и документирование в проектах разработки программного обеспечения. Не включает в себя детальную инструкцию по проектированию, не содержит описаний, как строить диаграммы на UML. Основная цель: эффективное моделирование и документирование; но не охватывает программирование и тестирование, не включает вопросы управления проектом, развёртывания и сопровождения системы. Однако включает в себя проверку модели кодом.

Agile Unified Process (AUP) упрощенная версия IBM Rational Unified Process (RUP), разработанная Скоттом Амблером, которая описывает простое и понятное приближение (модель) для создания программного обеспечения для бизнес-приложений.

Agile Data Method (англ.) — группа итеративных методов разработки программного обеспечения, в которых требования и решения достигаются в рамках сотрудничества разных кросс-функциональных команд.

DSDM основан на концепции быстрой разработки приложений (Rapid Application Development, RAD). Представляет собой итеративный и инкрементный подход, который придаёт особое значение продолжительному участию в процессе пользователя/потребителя.

Essential Unified Process (англ.) (EssUP).

Экстремальное программирование (англ. Extreme programming, XP).

Feature driven development (FDD) — функционально-ориентированная разработка. Используемое в FDD понятие функции или свойства (англ. feature) системы достаточно близко к понятию прецедента использования, используемому в RUP, существенное отличие — это дополнительное ограничение: «каждая функция должна допускать реализацию не более, чем за две недели». То есть если сценарий использования достаточно мал, его можно считать функцией. Если же велик, то его надо разбить на несколько относительно независимых функций.

Getting Real — итеративный подход без функциональных спецификаций, использующийся для веб-приложений. В данном методе сперва разрабатывается интерфейс программы, а потом её функциональная часть.

OpenUP — это итеративно-инкрементальный метод разработки программного обеспечения. Позиционируется как лёгкий и гибкий вариант RUP. OpenUP делит жизненный цикл проекта на четыре фазы: начальная фаза, фазы уточнения, конструирования и передачи. Жизненный цикл проекта обеспечивает предоставление заинтересованным лицам и членам коллектива точек ознакомления и принятия решений на протяжении всего проекта. Это позволяет эффективно контролировать ситуацию и вовремя принимать решения о приемлемости результатов. План проекта определяет жизненный цикл, а конечным результатом является окончательное приложение.

Scrum устанавливает правила управления процессом разработки и позволяет использовать уже существующие практики кодирования, корректируя требования или внося тактические изменения. Использование этой методологии дает возможность выявлять и устранять отклонения от желаемого результата на более ранних этапах разработки программного продукта.

Бережливая разработка программного обеспечения (англ. lean software development) использует подходы из концепции бережливого производства.

* 1. Обзор применяемых паттернов архитектуры ПО

Шаблон проектирования или паттерн (англ. design pattern) в разработке программного обеспечения — повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Обычно шаблон не является законченным образцом, который может быть прямо преобразован в код; это лишь пример решения задачи, который можно использовать в различных ситуациях. Объектно-ориентированные шаблоны показывают отношения и взаимодействия между классами или объектами, без определения того, какие конечные классы или объекты приложения будут использоваться.

«Низкоуровневые» шаблоны, учитывающие специфику конкретного языка программирования, называются идиомами. Это хорошие решения проектирования, характерные для конкретного языка или программной платформы, и потому не универсальные.

На наивысшем уровне существуют архитектурные шаблоны, они охватывают собой архитектуру всей программной системы.

Алгоритмы по своей сути также являются шаблонами, но не проектирования, а вычисления, так как решают вычислительные задачи.

1. Практическая часть
   1. Постановка задачи.

Приложение разработано в программной среде разработки Visual Studio Community 2019 .NET Framework 4.5 . Поставлена задача разработать консольное приложение для чтения и записи из файлов, для хранения данных о товарах, поставщика и поставках. Программа должна обеспечивать добавление, редактирование, удаление новых записей в базу данных через консоль и выводить готовые данные в документ Microsoft Excel. Программный продукт рассчитан на работу в сфере торговли.

* 1. Описание предметной области.

В соответствии с поставленной задачей образуются 3 обязательных класса: Товар – название товара, количество товара на складе, единица измерения товара, стоимость единицы товара, описание товара; Поставщик – название поставщика, адрес поставщика, номер телефона поставщика, ФИО контактного лица; Поставка – срок поставки, количество товаров в поставке, стоимость поставки.

Пользователь работает с проектом непосредственно через консоль, поэтому реализована примитивная работа с командами: пользователь вводит команду, если команда введена верно, то программа исполняет определенное действие, иначе выводится предупреждение. Программа имеет следующий набор команд:

Ввод – ввод данных через консоль;

Импорт – ввод данных из текстового файла;

Сохранить – сохранение данных в текстовый файл;

Экспорт – экспорт данных в документ Microsoft Excel;

Очистить – очищает базу данных программы;

Выход – выход из программы;

Отмена – доступна во время выполнения других команд, позволяет отменить выполнение команды;

2.3. Проектирование библиотеки классов.

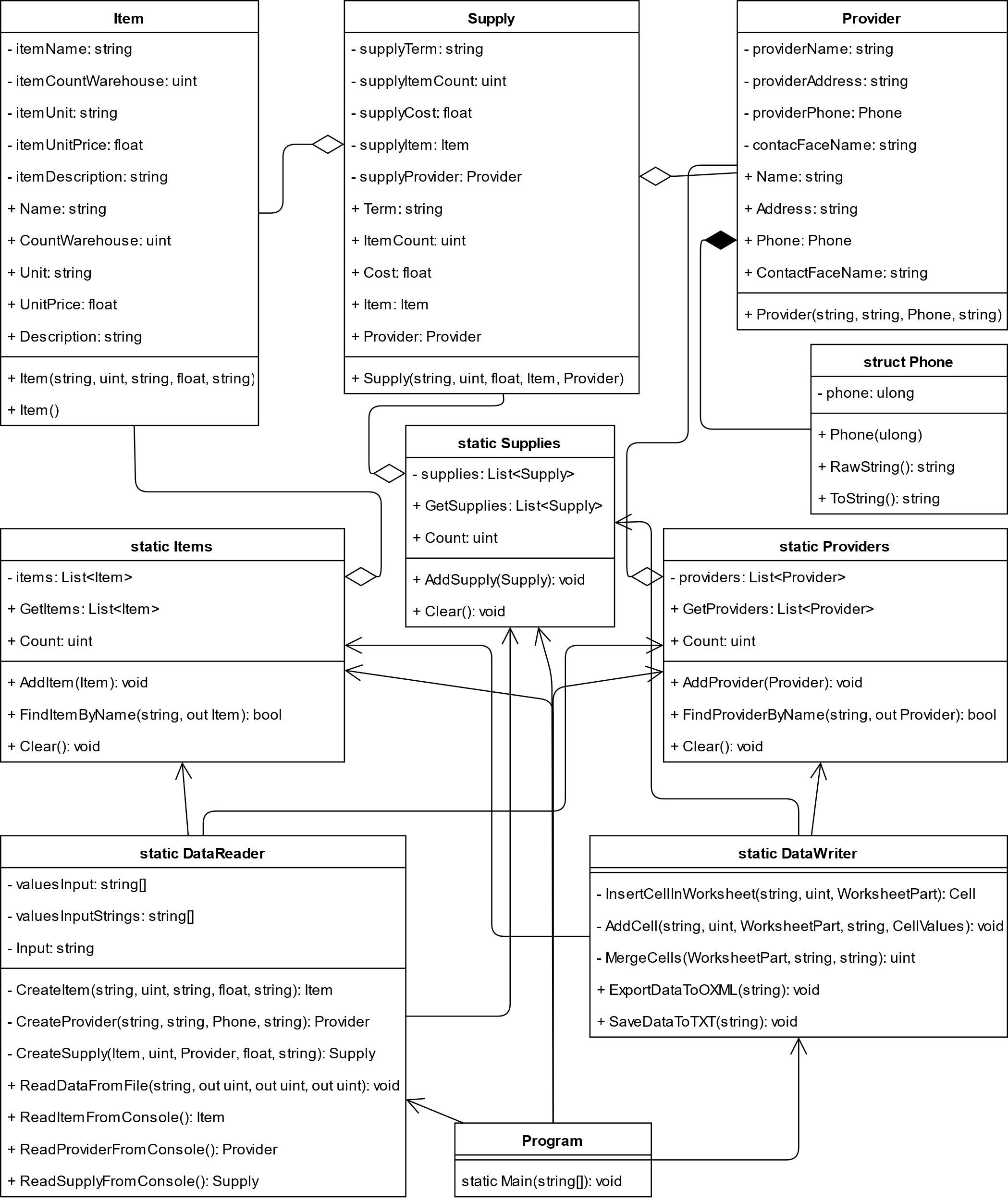


Рис. 1 Библиотека классов

2.4. Реализация библиотеки классов.

Товар реализован в виде класса Item.

Словесное описание работы класса:

1. Описание класса   
   *public class Item*
2. Объявление полей класса  *private string itemName;*

*private uint itemCountWarehouse  
private float itemUnitPrice;   
private string itemDescription;*

1. Объявление свойств с аксессорами get

*public string Name*

*{ get { return this.itemName; } }*

*public uint CountWarehouse*

*{ get { return this.itemCountWarehouse; } }*

*public string Unit*

*{ get { return this.itemUnit; } }*

*public float UnitPrice*

*{ get { return this.itemUnitPrice; } }*

*public string Description*

*{ get { return this.itemDescription; } }*

1. Объявление конструктора

*public Item(string name, uint countWarehouse, string unit, float unitPrice, string description)*

*{*

*this.itemName = name;*

*this.itemCountWarehouse = countWarehouse;*

*this.itemUnit = unit;*

*this.itemUnitPrice = unitPrice;*

*this.itemDescription = description;*

*}*

Поставщик реализован в виде класса Provider.

Словесное описание работы класса:

1. Описание класса  
   *public class Provider*
2. Описание полей класса  
   *private string providerName;*

*private string providerAddress;*

*private Phone providerPhone;*

*private string contactFaceName;*

1. Объявление свойств с аксессорами get  
   *public string Name*

*{ get { return this.providerName; } }*

*public string Address*

*{ get { return this.providerAddress; } }*

*public Phone Phone*

*{ get { return this.providerPhone; } }*

*public string ContactFaceName*

*{ get { return this.contactFaceName; } }*

1. Объявление конструктора  
   *public Provider(string name, string address, Phone phone, string contactFaceName)*

*{*

*this.providerName = name;*

*this.providerAddress = address;*

*this.providerPhone = phone;*

*this.contactFaceName = contactFaceName;*

*}*

Поставка реализована в виде класса Supply.

Словесное описание работы класса:

1. Описание класса  
   *public class Supply*
2. Описание полей класса  
   *private string supplyTerm;*

*private uint supplyItemCount;*

*private float supplyCost;*

*private Item supplyItem;*

*private Provider supplyProvider;*

1. Объявление свойств с аксессорами get  
   *public string Term*

*{ get { return this.supplyTerm; } }*

*public uint ItemCount*

*{ get { return this.supplyItemCount; } }*

*public float Cost*

*{ get { return this.supplyCost; } }*

*public Item Item*

*{ get { return this.supplyItem; } }*

*public Provider Provider*

*{ get { return this.supplyProvider; } }*

1. Объявление конструктора  
   *public Supply(string term, uint itemCount, float cost, Item item, Provider provider)*

*{*

*this.supplyTerm = term;*

*this.supplyItemCount = itemCount;*

*this.supplyCost = cost;*

*this.supplyItem = item;*

*this.supplyProvider = provider;*

*}*

Структура Phone хранит номер телефона в виде числа и содержит метод преобразования номера телефона в строку вида +X-(XXX)-XXX-XX-XX

Словесное описание работы структуры:

1. Описание структуры  
    *public struct Phone*
2. Описание поля структуры  
   *private ulong phone;*
3. Объявление конструктора   
   *public Phone(ulong number)*

*{*

Проверки на длину строки символов, которая может иметь длину от 11 до 15 символов

*if (number.ToString().Length > 15) throw new ArgumentException("Телефон должен содержать максимум 15 цифр.");*

*if (number.ToString().Length < 11) throw new ArgumentException("Телефон должен содержать как минимум 11 цифр.");*

*this.phone = number;*

*}*

1. Переопределение метода преобразования в строку  
   *public override string ToString()*

*{*

Инициализация переменных

*string phoneStr = "",   
numberStr = this.phone.ToString();*

*int countryCodeLength = -1 \* (10 - numberStr.Length);*

Прибавление символа к переменной phoneStr

*phoneStr += '+';* Прибавление к переменной phoneStr подстроки с заданной длиной

*phoneStr += numberStr.Substring(0, countryCodeLength);*

Прибавление к переменной phoneStr строки

*phoneStr += "-(";*Цикл с параметром i, от начального значения countryCodeLength до конечного значения i < countryCodeLength + 3 с шагом изменения 1

*for(int i = countryCodeLength; i <countryCodeLength + 3; i++)*

Прибавление к переменной phoneStr значения переменной numberStr[i]

*phoneStr += numberStr[i];*

Прибавление к переменной phoneStr строки

*phoneStr += ")-";*

Цикл с параметром i, от начального значения countryCodeLength + 3 до конечного значения i < countryCodeLength + 6 с шагом изменения 1

*for (int i = countryCodeLength +3; i <countryCodeLength +6; i++)*

Прибавление к переменной phoneStr значения переменной numberStr[i]

*phoneStr += numberStr[i];*

Прибавление символа к переменной phoneStr

*phoneStr += '-';*

Цикл с параметром i, от начального значения countryCodeLength + 6 до конечного значения i < countryCodeLength + 8 с шагом изменения 1

*for (int i = countryCodeLength +6; i <countryCodeLength +8; i ++)*

Прибавление к переменной phoneStr значения переменной numberStr[i]

*phoneStr += numberStr[i];*

Прибавление символа к переменной phoneStr

*phoneStr += '-';*

Цикл с параметром i, от начального значения countryCodeLength + 6 до конечного значения i < countryCodeLength + 8 с шагом изменения 1

*for (int i =countryCodeLength +8; i <countryCodeLength +10; i++)*

Прибавление к переменной phoneStr значения переменной numberStr[i]

*phoneStr += numberStr[i];*

Возврат значения переменной phoneStr

*return phoneStr;*

*}*

1. Метод преобразования числа в строку

*public string RawString()*

*{*

*return this.phone.ToString();*

*}*

Класс Items содержит в себе список всех товаров.

Словесное описание работы класса:

1. Описание класса  
   *public static class Items*
2. Объявление статического поля класса  
   *private static List<Item> items = new List<Item>();*
3. Объявление свойств с аксессорами get  
   *public static List<Item> GetItems*

*{ get { return items; } }*

*public static uint Count*

*{ get { return (uint)items.Count; } }*

1. Описание метода, который добавляет товар в список  
   *public static void AddItem(Item item)*

*{*

*items.Add(item);*

*}*

1. Объявление метода поиска товара по имени. При нахождении записывает товар в параметр item и возвращает истину.  
   *public static bool FindItemByName(string itemName, out Item item)*

*{*

*for (int i = 0; i < items.Count; i++)*

*{*

*if (items[i].Name == itemName)*

*{*

*item = items[i];*

*return true;*

*}*

*}*

*item = new Item();*

*return false;*

*}*

1. Объявление метода очистки списка.

*public static void Clear()*

*{*

*items.Clear();*

*}*

Класс Providers содержит в себе список всех поставщиков.

Словесное описание работы класса:

1. Описание класса  
   *public static class Providers*
2. Объявление статического поля класса  
   *private static List<Provider> providers = new List<Provider>();*
3. Объявление свойств с аксессорами get  
   *public static List<Provider> GetProviders*

*{ get { return providers; } }*

*public static uint Count*

*{ get { return (uint)providers.Count; } }*

1. Описание метода, который добавляет поставщика в список  
   *public static void AddProvider(Provider provider)*

*{*

*providers.Add(provider);*

*}*

1. Объявление метода поиска поставщика по имени. При нахождении записывает поставщика в параметр provider и возвращает истину.  
   *public static bool FindProviderByName(string providerName, out Provider provider)*

*{*

*for (int i = 0; i < providers.Count; i++)*

*{*

*if (providers[i].Name == providerName)*

*{*

*provider = providers[i];*

*return true;*

*}*

*}*

*provider = new Provider();*

*return false;*

*}*

1. Объявление метода очистки списка.

*public static void Clear()*

*{*

*items.Clear();*

*}*

Класс Supplies содержит в себе список всех поставок.

Словесное описание работы класса:

1. Описание класса  
   *public static class Supplies*
2. Объявление статического поля класса  
   *private static List<Supply> supplies = new List<Supply>();*
3. Объявление свойств с аксессорами get  
   *public static List<Supply> GetSupplies*

*{ get { return suppliess; } }*

*public static uint Count*

*{ get { return (uint)supplies.Count; } }*

1. Описание метода, который добавляет поставку в список  
   *public static void AddSupply(Supply supply)*

*{*

*supplies.Add(supply);*

*}*

1. Объявление метода очистки списка.

*public static void Clear()*

*{*

*supplies.Clear();*

*}*

2.5. Реализация методов работы с вводом-выводом.

Класс DataReader имеет в себе 4 метода для ввода. Первый метод позволяет считывать данные из файла построчно и делает проверку на введённые данные, если введены данные некорректно, то программа выводит предупреждение о некорректно введенных данных. Второй, третий и четвертый методы позволяют вводить данные через консоль, делая проверку на введённые данные, и если встречают некорректные данные, выводят «Данные введены некорректно, введите данные вновь». Данные вводятся в порядке:

1. Название товара
2. Единицы измерения товара
3. Стоимость единицы товара
4. Количество товара на складе
5. Описание товара
6. Название поставщика
7. Адрес поставщика
8. Телефон поставщика
9. ФИО контактного лица
10. Количество товаров в поставке
11. Стоимость поставки
12. Срок поставки

Класс DataWriter имеет в себе 2 метода для вывода. Первый метод реализует вывод данных в документ Microsoft Excel. Второй метод сохраняет данные в текстовый файл, который в дальнейшем может быть открыт программой.

1. Срок поставки
2. Количество товаров в поставке
3. Стоимость поставки
4. Название поставщика
5. Адрес поставщика
6. Телефон поставщика
7. ФИО контактного лица
8. Название товара
9. Количество товара на складе
10. Единица измерения товара
11. Стоимость единицы товара
12. Описание

Заключение.

При выполнении курсовой работы мной были закреплены навыки в программировании на С# в рамках изученного материала, освоены основные принципы разработки консольного приложения, были рассмотрены многие возможности работы Visual Studio Community 2019 и .NET Framework 4.5 .

Стоит отметить, что на данный момент функционала приложения теоретически достаточно для того, чтобы человек знающий основы работы в консоли смог использовать этот продукт в масштабах оптовой базы.

На данный момент пользователь может вводить данные как вручную через консоль, так и через готовые текстовые документы, выводить данные в документ Microsoft Excel и сохранять данные в текстовый файл для дальнейшего использования их в программе.

Планируемые усовершенствования данной курсовой работы:

* Добавление визуальной части, графического интерфейса минимум через Windows Forms;
* Добавление поиска по различным критериям;
* Перемещение базы данных на сервер для доступа к информации через разные устройства, но с индивидуальными паролями для доступа и редактирования данных.

Список литературы.

1. Мартин Р., Мартин М. Принципы, паттерны и методики гибкой разработки на языке C# - Символ-Плюс, 2011;
2. Роберт С. Мартин, Джеймс В. Ньюкирк, Роберт С. Косс Быстрая разработка программ. Принципы, примеры, практика – Вильямс, 2004;
3. Герберт Шилдт "C# 4.0. Полное руководство";
4. Макконнелл "Совершенный код. Практическое руководство по разработке программного обеспечения";
5. Обзор паттернов проектирования (<http://citforum.ru/SE/project/pattern/>);
6. 12 методологий разработки ПО (<https://geekbrains.ru/posts/methodologies>);
7. Язык C# и платформа .NET Core (<https://metanit.com/sharp/tutorial/1.1.php>);
8. DocumentFormat.OpenXml Namespace (<https://docs.microsoft.com/ru-ru/dotnet/api/documentformat.openxml?view=openxml-2.8.1>);
9. Руководство по .NET (<https://docs.microsoft.com/ru-ru/dotnet/standard/>);
10. Гибкая методология разработки (<https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%B1%D0%BA%D0%B0%D1%8F_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8>);
11. Microsoft Excel (<https://ru.wikipedia.org/wiki/Microsoft_Excel>).

Приложение 1.

Реализация класса ввода

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Text;*

*using System.Threading.Tasks;*

*using System.IO;*

*namespace WholesaleBase*

*{*

*public static class DataReader*

*{*

*private static string[] valuesInput = { "itemName", "itemUnit", "itemUnitPrice", "itemCountWarehouse", "itemDescription",*

*"providerName", "providerAddress", "providerPhone", "contactFaceName", "supplyItemCount", "supplyCost", "supplyTerm"};*

*private static string[] valuesInputStrings = { "Название товара:", "Единица измерения товара:", "Стоимость единицы товара:",*

*"Количество товара на складе:", "Описание товара (Может быть пустым):", "Название поставщика:", "Адрес поставщика:",*

*"Телефон поставщика:", "ФИО контактного лица:", "Количество товаров в поставке:", "Стоимость поставки:", "Срок поставки"};*

*private static Item CreateItem(string itemName, uint itemCountWarehouse, string itemUnit, float itemUnitPrice, string itemDescription)*

*{*

*return new Item(itemName, itemCountWarehouse, itemUnit, itemUnitPrice, itemDescription);*

*}*

*private static Provider CreateProvider(string providerName, string providerAddress, Phone providerPhone, string contactFaceName)*

*{*

*return new Provider(providerName, providerAddress, providerPhone, contactFaceName);*

*}*

*private static Supply CreateSupply(Item supplyItem, uint supplyItemCount, Provider supplyProvider, float supplyCost, string supplyTerm)*

*{*

*return new Supply(supplyTerm, supplyItemCount, supplyCost, supplyItem, supplyProvider);*

*}*

*public static void ReadDataFromFile(string path, out uint items, out uint providers, out uint supplies)*

*{*

*items = 0;*

*providers = 0;*

*supplies = 0;*

*FileInfo file;*

*StreamReader reader;*

*file = new FileInfo(path);*

*if (!file.Exists) throw new Exception("Невозможно открыть файл.");*

*reader = new StreamReader(file.OpenRead(), Encoding.Default);*

*bool itemExists = false, providerExists = false;*

*while (!reader.EndOfStream)*

*{*

*switch(reader.ReadLine().ToUpper())*

*{*

*case "ТОВАР":*

*{*

*string itemName = reader.ReadLine();*

*if (itemName.StartsWith("\n") || itemName.StartsWith(" ") || itemName.Length == 0)*

*{*

*Console.WriteLine("Некорректно указано название товара.");*

*}*

*string itemUnit = reader.ReadLine();*

*if (itemUnit.StartsWith("\n") || itemUnit.StartsWith(" ") || itemUnit.Length == 0)*

*{*

*Console.WriteLine("Некорректно указана единица измерения товара {0}.", itemName);*

*}*

*float itemUnitPrice;*

*bool successUnitPrice = float.TryParse(reader.ReadLine(), out itemUnitPrice);*

*if (!successUnitPrice)*

*{*

*itemUnitPrice = 0;*

*Console.WriteLine("Некорректно указана стоимость единицы товара {0}.", itemName);*

*}*

*uint itemCountWarehouse;*

*bool successCountWarehouse = uint.TryParse(reader.ReadLine(), out itemCountWarehouse);*

*if (!successCountWarehouse)*

*{*

*itemCountWarehouse = 0;*

*Console.WriteLine("Некорректно указано количество товара {0} на складе.", itemName);*

*}*

*string itemDescription = reader.ReadLine();*

*Items.AddItem(CreateItem(itemName, itemCountWarehouse, itemUnit, itemUnitPrice, itemDescription));*

*items++;*

*break;*

*}*

*case "ПОСТАВЩИК":*

*{*

*string providerName = reader.ReadLine();*

*if (providerName.StartsWith("\n") || providerName.StartsWith(" ") || providerName.Length == 0)*

*{*

*Console.WriteLine("Некорректно указано название поставщика.");*

*}*

*string providerAddress = reader.ReadLine();*

*if (providerAddress.StartsWith("\n") || providerAddress.StartsWith(" ") || providerAddress.Length == 0)*

*{*

*Console.WriteLine("Некорректно указан адрес поставщика {0}.", providerName);*

*}*

*Phone providerPhone;*

*ulong number;*

*bool successNumber = ulong.TryParse(reader.ReadLine(), out number);*

*if (!successNumber)*

*{*

*number = 0;*

*Console.WriteLine("Некорректно указан номер телефона поставщика {0}", providerName);*

*}*

*try*

*{*

*providerPhone = new Phone(number);*

*}*

*catch (Exception e)*

*{*

*Console.WriteLine("Ошибка в поставщике {1}:\n{0}", e.Message, providerName);*

*providerPhone = new Phone(0);*

*}*

*string contactFaceName = reader.ReadLine();*

*if (contactFaceName.StartsWith("\n") || contactFaceName.StartsWith(" ") || contactFaceName.Length == 0)*

*{*

*Console.WriteLine("Некорректно указано имя контактного лица поставщика {0}.", providerName);*

*}*

*Providers.AddProvider(CreateProvider(providerName, providerAddress, providerPhone, contactFaceName));*

*providers++;*

*break;*

*}*

*case "ПОСТАВКА":*

*{*

*string itemName = reader.ReadLine();*

*if (itemName.StartsWith("\n") || itemName.StartsWith(" ") || itemName.Length == 0)*

*{*

*Console.WriteLine("Некорректно указано название товара.");*

*}*

*Item item;*

*if (!Items.FindItemByName(itemName, out item) || !Char.IsDigit((char)reader.Peek()))*

*{*

*string itemUnit = reader.ReadLine();*

*if (itemUnit.StartsWith("\n") || itemUnit.StartsWith(" ") || itemUnit.Length == 0)*

*{*

*Console.WriteLine("Некорректно указана единица измерения товара {0}.", itemName);*

*}*

*float itemUnitPrice;*

*bool successUnitPrice = float.TryParse(reader.ReadLine(), out itemUnitPrice);*

*if (!successUnitPrice)*

*{*

*itemUnitPrice = 0;*

*Console.WriteLine("Некорректно указана стоимость единицы товара {0}.", itemName);*

*}*

*uint itemCountWarehouse;*

*bool successCountWarehouse = uint.TryParse(reader.ReadLine(), out itemCountWarehouse);*

*if (!successCountWarehouse)*

*{*

*itemCountWarehouse = 0;*

*Console.WriteLine("Некорректно указано количество товара {0} на складе.", itemName);*

*}*

*string itemDescription = reader.ReadLine();*

*item = CreateItem(itemName, itemCountWarehouse, itemUnit, itemUnitPrice, itemDescription);*

*Items.AddItem(item);*

*items++;*

*}*

*uint supplyItemCount;*

*bool successSupplyItemCount = uint.TryParse(reader.ReadLine(), out supplyItemCount);*

*if (!successSupplyItemCount)*

*{*

*supplyItemCount = 0;*

*Console.WriteLine("Некорректно указано количество товара {0} в поставке.", itemName);*

*}*

*string providerName = reader.ReadLine();*

*if (providerName.StartsWith("\n") || providerName.StartsWith(" ") || providerName.Length == 0)*

*{*

*Console.WriteLine("Некорректно указано название поставщика.");*

*}*

*Provider provider;*

*if(!Providers.FindProviderByName(providerName, out provider) || !Char.IsDigit((char)reader.Peek()))*

*{*

*string providerAddress = reader.ReadLine();*

*if (providerAddress.StartsWith("\n") || providerAddress.StartsWith(" ") || providerAddress.Length == 0)*

*{*

*Console.WriteLine("Некорректно указан адрес поставщика {0}.", providerName);*

*}*

*Phone providerPhone;*

*ulong number;*

*bool successNumber = ulong.TryParse(reader.ReadLine(), out number);*

*if (!successNumber)*

*{*

*number = 0;*

*Console.WriteLine("Некорректно указан номер телефона поставщика {0}", providerName);*

*}*

*try*

*{*

*providerPhone = new Phone(number);*

*}*

*catch (Exception e)*

*{*

*Console.WriteLine("Ошибка в поставщике {1}:\n{0}", e.Message, providerName);*

*providerPhone = new Phone(0);*

*}*

*string contactFaceName = reader.ReadLine();*

*if (contactFaceName.StartsWith("\n") || contactFaceName.StartsWith(" ") || contactFaceName.Length == 0)*

*{*

*Console.WriteLine("Некорректно указано имя контактного лица поставщика {0}.", providerName);*

*}*

*provider = CreateProvider(providerName, providerAddress, providerPhone, contactFaceName);*

*Providers.AddProvider(provider);*

*providers++;*

*}*

*float supplyCost;*

*bool successSupplyCost = float.TryParse(reader.ReadLine(), out supplyCost);*

*if (!successSupplyCost)*

*{*

*supplyCost = 0;*

*Console.WriteLine("Некорректно указана стоимость поставки товара {0} от поставщика {1}.", itemName, providerName);*

*}*

*string supplyTerm = reader.ReadLine();*

*if (supplyTerm.StartsWith("\n") || supplyTerm.StartsWith(" ") || supplyTerm.Length == 0)*

*{*

*Console.WriteLine("Некорректно указан срок поставки товара {0} от поставщика {1}.", itemName, providerName);*

*}*

*Supply supply = CreateSupply(item, supplyItemCount, provider, supplyCost, supplyTerm);*

*Supplies.AddSupply(supply);*

*supplies++;*

*break;*

*}*

*default:*

*{*

*break;*

*}*

*}*

*}*

*}*

*private static string Input*

*{*

*get*

*{*

*string input = Console.ReadLine();*

*if (input.ToUpper() == "ОТМЕНА") throw new Exception("Отмена ввода.");*

*else return input;*

*}*

*}*

*public static Item ReadItemFromConsole()*

*{*

*string itemName = "";*

*string itemUnit = "";*

*float itemUnitPrice = 0;*

*uint itemCountWarehouse = 0;*

*string itemDescription = "";*

*Console.WriteLine("\"отмена\" - отмена ввода.");*

*for (int i = 0; i < 5; i++)*

*{*

*Console.WriteLine(valuesInputStrings[i]);*

*switch (valuesInput[i])*

*{*

*case "itemName":*

*{*

*itemName = Input;*

*while (itemName.StartsWith("\n") || itemName.StartsWith(" ") || itemName.Length == 0)*

*{*

*Console.WriteLine("Название товара не должно начинаться с пробела или быть пустым. Повторите ввод.");*

*itemName = Input;*

*}*

*break;*

*}*

*case "itemUnit":*

*{*

*itemUnit = Input;*

*while (itemUnit.StartsWith("\n") || itemUnit.StartsWith(" ") || itemUnit.Length == 0)*

*{*

*Console.WriteLine("Единица измерения товара не должна начинаться с пробела или быть пустой. Повторите ввод.");*

*itemUnit = Input;*

*}*

*break;*

*}*

*case "itemUnitPrice":*

*{*

*bool successUnitPrice = float.TryParse(Input, out itemUnitPrice);*

*while (!successUnitPrice)*

*{*

*Console.WriteLine("Некорректно указана цена единицы товара. Повторите ввод.");*

*successUnitPrice = float.TryParse(Input, out itemUnitPrice);*

*}*

*break;*

*}*

*case "itemCountWarehouse":*

*{*

*bool successCountWarehouse = uint.TryParse(Input, out itemCountWarehouse);*

*while (!successCountWarehouse)*

*{*

*Console.WriteLine("Некорректно указано количество товара на складе. Повторите ввод.");*

*successCountWarehouse = uint.TryParse(Input, out itemCountWarehouse);*

*}*

*break;*

*}*

*case "itemDescription":*

*{*

*itemDescription = Input;*

*break;*

*}*

*default:*

*{*

*break;*

*}*

*}*

*}*

*Item item = CreateItem(itemName, itemCountWarehouse, itemUnit, itemUnitPrice, itemDescription);*

*return item;*

*}*

*public static Provider ReadProviderFromConsole()*

*{*

*string providerName = "";*

*string providerAddress = "";*

*Phone providerPhone = new Phone();*

*string contactFaceName = "";*

*Console.WriteLine("\"отмена\" - отмена ввода.");*

*for (int i = 5; i < 9; i++)*

*{*

*Console.WriteLine(valuesInputStrings[i]);*

*switch (valuesInput[i])*

*{*

*case "providerName":*

*{*

*providerName = Input;*

*while (providerName.StartsWith("\n") || providerName.StartsWith(" ") || providerName.Length == 0)*

*{*

*Console.WriteLine("Название производителя не должно начинаться с пробела или быть пустым. Повторите ввод.");*

*providerName = Input;*

*}*

*break;*

*}*

*case "providerAddress":*

*{*

*providerAddress = Input;*

*while (providerAddress.StartsWith("\n") || providerAddress.StartsWith(" ") || providerAddress.Length == 0)*

*{*

*Console.WriteLine("Адрес помтавщика не может начинаться с пробела или быть пустым. Повторите ввод.");*

*providerAddress = Input;*

*}*

*break;*

*}*

*case "providerPhone":*

*{*

*ulong number;*

*bool successNumber = ulong.TryParse(Input, out number);*

*try*

*{*

*providerPhone = new Phone(number);*

*successNumber = true;*

*}*

*catch (Exception e)*

*{*

*Console.WriteLine(e.Message);*

*successNumber = false;*

*}*

*while (!successNumber)*

*{*

*Console.WriteLine("Некорректно указан номер телефона поставщика. Повторите ввод.");*

*successNumber = ulong.TryParse(Input, out number);*

*try*

*{*

*providerPhone = new Phone(number);*

*successNumber = true;*

*}*

*catch (Exception e)*

*{*

*Console.WriteLine(e.Message);*

*Console.WriteLine("Повторите ввод.");*

*successNumber = false;*

*}*

*}*

*break;*

*}*

*case "contactFaceName":*

*{*

*contactFaceName = Input;*

*while (contactFaceName.StartsWith("\n") || contactFaceName.StartsWith(" ") || contactFaceName.Length == 0)*

*{*

*Console.WriteLine("Имя контактного лица не должно быть пустым или начинаться с пробела. Повторите ввод.");*

*contactFaceName = Input;*

*}*

*break;*

*}*

*default:*

*{*

*break;*

*}*

*}*

*}*

*Provider provider = CreateProvider(providerName, providerAddress, providerPhone, contactFaceName);*

*return provider;*

*}*

*public static Supply ReadSupplyFromConsole()*

*{*

*string itemName = "";*

*string itemUnit = "";*

*float itemUnitPrice = 0;*

*uint itemCountWarehouse = 0;*

*string itemDescription = "";*

*string providerName = "";*

*string providerAddress = "";*

*Phone providerPhone = new Phone();*

*string contactFaceName = "";*

*string supplyTerm = "";*

*uint supplyItemCount = 0;*

*float supplyCost = 0;*

*bool itemExists = false, providerExists = false;*

*Item item = new Item();*

*Provider provider = new Provider();*

*Console.WriteLine("\"отмена\" - отмена ввода.");*

*for (int i = 0; i < valuesInput.Length; i++)*

*{*

*Console.WriteLine(valuesInputStrings[i]);*

*switch (valuesInput[i])*

*{*

*case "itemName":*

*{*

*itemName = Input;*

*while (itemName.StartsWith("\n") || itemName.StartsWith(" ") || itemName.Length == 0)*

*{*

*Console.WriteLine("Название товара не должно начинаться с пробела или быть пустым. Повторите ввод.");*

*itemName = Input;*

*}*

*if(Items.FindItemByName(itemName, out item))*

*{*

*Console.WriteLine("Товар {0} уже существует. Продолжить ввод? \"да\" - продолжить, остальное - подстановка значения из базы.", itemName);*

*if (!(Console.ReadLine().ToUpper() == "ДА"))*

*{*

*itemExists = true;*

*i += 4;*

*}*

*}*

*break;*

*}*

*case "itemUnit":*

*{*

*itemUnit = Input;*

*while (itemUnit.StartsWith("\n") || itemUnit.StartsWith(" ") || itemUnit.Length == 0)*

*{*

*Console.WriteLine("Единица измерения товара не должна начинаться с пробела или быть пустой. Повторите ввод.");*

*itemUnit = Input;*

*}*

*break;*

*}*

*case "itemUnitPrice":*

*{*

*bool successUnitPrice = float.TryParse(Input, out itemUnitPrice);*

*while (!successUnitPrice)*

*{*

*Console.WriteLine("Некорректно указана цена единицы товара. Повторите ввод.");*

*successUnitPrice = float.TryParse(Input, out itemUnitPrice);*

*}*

*break;*

*}*

*case "itemCountWarehouse":*

*{*

*bool successCountWarehouse = uint.TryParse(Input, out itemCountWarehouse);*

*while (!successCountWarehouse)*

*{*

*Console.WriteLine("Некорректно указано количество товара на складе. Повторите ввод.");*

*successCountWarehouse = uint.TryParse(Input, out itemCountWarehouse);*

*}*

*break;*

*}*

*case "itemDescription":*

*{*

*itemDescription = Input;*

*break;*

*}*

*case "providerName":*

*{*

*providerName = Input;*

*while (providerName.StartsWith("\n") || providerName.StartsWith(" ") || providerName.Length == 0)*

*{*

*Console.WriteLine("Название производителя не должно начинаться с пробела или быть пустым. Повторите ввод.");*

*providerName = Input;*

*}*

*if (Providers.FindProviderByName(providerName, out provider))*

*{*

*Console.WriteLine("Поставщик {0} уже существует. Продолжить ввод? \"да\" - продолжить, остальное - подстановка значения из базы.", providerName);*

*if (!(Console.ReadLine().ToUpper() == "ДА"))*

*{*

*providerExists = true;*

*i += 3;*

*}*

*}*

*break;*

*}*

*case "providerAddress":*

*{*

*providerAddress = Input;*

*while (providerAddress.StartsWith("\n") || providerAddress.StartsWith(" ") || providerAddress.Length == 0)*

*{*

*Console.WriteLine("Адрес помтавщика не может начинаться с пробела или быть пустым. Повторите ввод.");*

*providerAddress = Input;*

*}*

*break;*

*}*

*case "providerPhone":*

*{*

*ulong number;*

*bool successNumber = ulong.TryParse(Input, out number);*

*try*

*{*

*providerPhone = new Phone(number);*

*successNumber = true;*

*}*

*catch (Exception e)*

*{*

*Console.WriteLine(e.Message);*

*successNumber = false;*

*}*

*while (!successNumber)*

*{*

*Console.WriteLine("Некорректно указан номер телефона поставщика. Повторите ввод.");*

*successNumber = ulong.TryParse(Input, out number);*

*try*

*{*

*providerPhone = new Phone(number);*

*successNumber = true;*

*}*

*catch (Exception e)*

*{*

*Console.WriteLine(e.Message);*

*Console.WriteLine("Повторите ввод.");*

*successNumber = false;*

*}*

*}*

*break;*

*}*

*case "contactFaceName":*

*{*

*contactFaceName = Input;*

*while (contactFaceName.StartsWith("\n") || contactFaceName.StartsWith(" ") || contactFaceName.Length == 0)*

*{*

*Console.WriteLine("Имя контактного лица не должно быть пустым или начинаться с пробела. Повторите ввод.");*

*contactFaceName = Input;*

*}*

*break;*

*}*

*case "supplyTerm":*

*{*

*supplyTerm = Input;*

*while (supplyTerm.StartsWith("\n") || supplyTerm.StartsWith(" ") || supplyTerm.Length == 0)*

*{*

*Console.WriteLine("Срок поставки не должен начинаться с пробела или быть пустым. Повторите ввод.");*

*supplyTerm = Input;*

*}*

*break;*

*}*

*case "supplyItemCount":*

*{*

*bool successSupplyItemCount = uint.TryParse(Input, out supplyItemCount);*

*while (!successSupplyItemCount)*

*{*

*Console.WriteLine("Некорректно указано количество товара в поставке. Повторите ввод.");*

*successSupplyItemCount = uint.TryParse(Input, out supplyItemCount);*

*}*

*break;*

*}*

*case "supplyCost":*

*{*

*bool successSupplyCost = float.TryParse(Input, out supplyCost);*

*while (!successSupplyCost)*

*{*

*Console.WriteLine("Некорректно указана стоимость поставки. Повторите ввод.");*

*successSupplyCost = float.TryParse(Input, out supplyCost);*

*}*

*break;*

*}*

*default:*

*{*

*break;*

*}*

*}*

*}*

*if (!itemExists)*

*{*

*item = CreateItem(itemName, itemCountWarehouse, itemUnit, itemUnitPrice, itemDescription);*

*Items.AddItem(item);*

*}*

*if (!providerExists)*

*{*

*provider = CreateProvider(providerName, providerAddress, providerPhone, contactFaceName);*

*Providers.AddProvider(provider);*

*}*

*Supply supply = CreateSupply(item, supplyItemCount, provider, supplyCost, supplyTerm);*

*return supply;*

*}*

*}*

*}*

Приложение 2.

Реализация класса вывода

*using System;*

*using System.Collections.Generic;*

*using System.Linq;*

*using System.Text;*

*using System.Threading.Tasks;*

*using System.IO;*

*using DocumentFormat.OpenXml;*

*using DocumentFormat.OpenXml.Packaging;*

*using DocumentFormat.OpenXml.Spreadsheet;*

*namespace WholesaleBase*

*{*

*public static class DataWriter*

*{*

*private static Cell InsertCellInWorksheet(string columnName, uint rowIndex, WorksheetPart worksheetPart)*

*{*

*Worksheet worksheet = worksheetPart.Worksheet;*

*SheetData sheetData = worksheet.GetFirstChild<SheetData>();*

*string cellReference = columnName + rowIndex;*

*// If the worksheet does not contain a row with the specified row index, insert one.*

*Row row;*

*if (sheetData.Elements<Row>().Where(r => r.RowIndex == rowIndex).Count() != 0)*

*{*

*row = sheetData.Elements<Row>().Where(r => r.RowIndex == rowIndex).First();*

*}*

*else*

*{*

*row = new Row() { RowIndex = rowIndex };*

*sheetData.Append(row);*

*}*

*// If there is not a cell with the specified column name, insert one.*

*if (row.Elements<Cell>().Where(c => c.CellReference.Value == columnName + rowIndex).Count() > 0)*

*{*

*return row.Elements<Cell>().Where(c => c.CellReference.Value == cellReference).First();*

*}*

*else*

*{*

*// Cells must be in sequential order according to CellReference. Determine where to insert the new cell.*

*Cell refCell = null;*

*foreach (Cell cell in row.Elements<Cell>())*

*{*

*if (cell.CellReference.Value.Length == cellReference.Length)*

*{*

*if (string.Compare(cell.CellReference.Value, cellReference, true) > 0)*

*{*

*refCell = cell;*

*break;*

*}*

*}*

*}*

*Cell newCell = new Cell() { CellReference = cellReference };*

*row.InsertBefore(newCell, refCell);*

*worksheet.Save();*

*return newCell;*

*}*

*}*

*private static void AddCell(string column, uint row, WorksheetPart worksheetPart, string value, CellValues dataType)*

*{*

*Worksheet worksheet = worksheetPart.Worksheet;*

*Cell cell = InsertCellInWorksheet(column, row, worksheetPart);*

*cell.CellValue = new CellValue(value);*

*cell.DataType = dataType;*

*worksheet.Save();*

*}*

*private static void MergeCells(WorksheetPart worksheetPart, string cell1Name, string cell2Name)*

*{*

*Worksheet worksheet = worksheetPart.Worksheet;*

*MergeCells mergeCells;*

*if (worksheet.Elements<MergeCells>().Count() > 0)*

*mergeCells = worksheet.Elements<MergeCells>().First();*

*else*

*{*

*mergeCells = new MergeCells();*

*// Insert a MergeCells object into the specified position.*

*if (worksheet.Elements<CustomSheetView>().Count() > 0)*

*worksheet.InsertAfter(mergeCells, worksheet.Elements<CustomSheetView>().First());*

*else*

*worksheet.InsertAfter(mergeCells, worksheet.Elements<SheetData>().First());*

*}*

*// Create the merged cell and append it to the MergeCells collection.*

*MergeCell mergeCell = new MergeCell()*

*{*

*Reference =*

*new StringValue(cell1Name + ":" + cell2Name)*

*};*

*mergeCells.Append(mergeCell);*

*worksheet.Save();*

*}*

*public static void ExportDataToOXML(string fileName)*

*{*

*SpreadsheetDocument document;*

*try*

*{*

*document = SpreadsheetDocument.Create(fileName, SpreadsheetDocumentType.Workbook);*

*}*

*catch (Exception e)*

*{*

*Console.WriteLine(e.Message);*

*throw new Exception("Невозможно создать файл.");*

*}*

*WorkbookPart workbookPart = document.AddWorkbookPart();*

*workbookPart.Workbook = new Workbook();*

*WorksheetPart worksheetPart = workbookPart.AddNewPart<WorksheetPart>();*

*worksheetPart.Worksheet = new Worksheet(new SheetData());*

*WorksheetPart worksheetPart2 = workbookPart.AddNewPart<WorksheetPart>();*

*worksheetPart2.Worksheet = new Worksheet(new SheetData());*

*WorksheetPart worksheetPart3 = workbookPart.AddNewPart<WorksheetPart>();*

*worksheetPart3.Worksheet = new Worksheet(new SheetData());*

*Sheets sheets = document.WorkbookPart.Workbook.AppendChild<Sheets>(new Sheets());*

*Sheet sheet = new Sheet() { Id = document.WorkbookPart.GetIdOfPart(worksheetPart), SheetId = 1, Name = "Товары" };*

*sheets.Append(sheet);*

*Sheet sheet2 = new Sheet() { Id = document.WorkbookPart.GetIdOfPart(worksheetPart2), SheetId = 2, Name = "Поставщики" };*

*sheets.Append(sheet2);*

*Sheet sheet3 = new Sheet() { Id = document.WorkbookPart.GetIdOfPart(worksheetPart3), SheetId = 3, Name = "Поставки" };*

*sheets.Append(sheet3);*

*AddCell("A", 1, worksheetPart, "Название", CellValues.String);*

*AddCell("B", 1, worksheetPart, "Количество на складе", CellValues.String);*

*AddCell("C", 1, worksheetPart, "Единица измерения", CellValues.String);*

*AddCell("D", 1, worksheetPart, "Стоимость единицы", CellValues.String);*

*AddCell("E", 1, worksheetPart, "Описание", CellValues.String);*

*for (int i = 0; i < Items.Count; i++)*

*{*

*AddCell("A", (uint)i + 2, worksheetPart, Items.GetItems[i].Name, CellValues.String);*

*AddCell("B", (uint)i + 2, worksheetPart, Items.GetItems[i].CountWarehouse.ToString(), CellValues.Number);*

*AddCell("C", (uint)i + 2, worksheetPart, Items.GetItems[i].Unit, CellValues.String);*

*AddCell("D", (uint)i + 2, worksheetPart, Items.GetItems[i].UnitPrice.ToString(), CellValues.Number);*

*AddCell("E", (uint)i + 2, worksheetPart, Items.GetItems[i].Description, CellValues.String);*

*}*

*AddCell("A", 1, worksheetPart2, "Название", CellValues.String);*

*AddCell("B", 1, worksheetPart2, "Адрес", CellValues.String);*

*AddCell("C", 1, worksheetPart2, "Телефон", CellValues.String);*

*AddCell("D", 1, worksheetPart2, "ФИО контактного лица", CellValues.String);*

*for (int i = 0; i < Providers.Count; i++)*

*{*

*AddCell("A", (uint)i + 2, worksheetPart2, Providers.GetProviders[i].Name, CellValues.String);*

*AddCell("B", (uint)i + 2, worksheetPart2, Providers.GetProviders[i].Address, CellValues.String);*

*AddCell("C", (uint)i + 2, worksheetPart2, Providers.GetProviders[i].Phone.ToString(), CellValues.String);*

*AddCell("D", (uint)i + 2, worksheetPart2, Providers.GetProviders[i].ContactFaceName, CellValues.String);*

*}*

*AddCell("A", 1, worksheetPart3, "Детали поставки", CellValues.String);*

*MergeCells(worksheetPart3, "A1", "C1");*

*AddCell("D", 1, worksheetPart3, "Поставщик", CellValues.String);*

*MergeCells(worksheetPart3, "D1", "G1");*

*AddCell("H", 1, worksheetPart3, "Товар", CellValues.String);*

*MergeCells(worksheetPart3, "H1", "L1");*

*AddCell("A", 2, worksheetPart3, "Срок поставки", CellValues.String);*

*AddCell("B", 2, worksheetPart3, "Количество товаров", CellValues.String);*

*AddCell("C", 2, worksheetPart3, "Стоимость поставки", CellValues.String);*

*AddCell("D", 2, worksheetPart3, "Название поставщика", CellValues.String);*

*AddCell("E", 2, worksheetPart3, "Адрес", CellValues.String);*

*AddCell("F", 2, worksheetPart3, "Телефон", CellValues.String);*

*AddCell("G", 2, worksheetPart3, "ФИО контактного лица", CellValues.String);*

*AddCell("H", 2, worksheetPart3, "Название товара", CellValues.String);*

*AddCell("I", 2, worksheetPart3, "Количество на складе", CellValues.String);*

*AddCell("J", 2, worksheetPart3, "Единица измерения", CellValues.String);*

*AddCell("K", 2, worksheetPart3, "Стоимость единицы", CellValues.String);*

*AddCell("L", 2, worksheetPart3, "Описание", CellValues.String);*

*for (int i = 0; i < Supplies.Count; i++)*

*{*

*AddCell("A", (uint)i + 3, worksheetPart3, Supplies.GetSupplies[i].Term, CellValues.String);*

*AddCell("B", (uint)i + 3, worksheetPart3, Supplies.GetSupplies[i].ItemCount.ToString(), CellValues.Number);*

*AddCell("C", (uint)i + 3, worksheetPart3, Supplies.GetSupplies[i].Cost.ToString(), CellValues.Number);*

*AddCell("D", (uint)i + 3, worksheetPart3, Supplies.GetSupplies[i].Provider.Name, CellValues.String);*

*AddCell("E", (uint)i + 3, worksheetPart3, Supplies.GetSupplies[i].Provider.Address, CellValues.String);*

*AddCell("F", (uint)i + 3, worksheetPart3, Supplies.GetSupplies[i].Provider.Phone.ToString(), CellValues.String);*

*AddCell("G", (uint)i + 3, worksheetPart3, Supplies.GetSupplies[i].Provider.ContactFaceName, CellValues.String);*

*AddCell("H", (uint)i + 3, worksheetPart3, Supplies.GetSupplies[i].Item.Name, CellValues.String);*

*AddCell("I", (uint)i + 3, worksheetPart3, Supplies.GetSupplies[i].Item.CountWarehouse.ToString(), CellValues.Number);*

*AddCell("J", (uint)i + 3, worksheetPart3, Supplies.GetSupplies[i].Item.Unit, CellValues.String);*

*AddCell("K", (uint)i + 3, worksheetPart3, Supplies.GetSupplies[i].Item.UnitPrice.ToString(), CellValues.String);*

*AddCell("L", (uint)i + 3, worksheetPart3, Supplies.GetSupplies[i].Item.Description, CellValues.String);*

*}*

*workbookPart.Workbook.Save();*

*document.Close();*

*}*

*public static void SaveDataToTXT(string fileName)*

*{*

*StreamWriter sw;*

*try*

*{*

*sw = new StreamWriter(File.Create(fileName), Encoding.Default);*

*}*

*catch (Exception e)*

*{*

*Console.WriteLine(e.Message);*

*throw new Exception("Невозможно создать файл.");*

*}*

*for(int i = 0; i < Items.Count; i++)*

*{*

*sw.WriteLine("ТОВАР");*

*sw.WriteLine(Items.GetItems[i].Name);*

*sw.WriteLine(Items.GetItems[i].Unit);*

*sw.WriteLine(Items.GetItems[i].UnitPrice.ToString());*

*sw.WriteLine(Items.GetItems[i].CountWarehouse.ToString());*

*sw.WriteLine(Items.GetItems[i].Description);*

*sw.WriteLine();*

*}*

*for (int i = 0; i < Providers.Count; i++)*

*{*

*sw.WriteLine("ПОСТАВЩИК");*

*sw.WriteLine(Providers.GetProviders[i].Name);*

*sw.WriteLine(Providers.GetProviders[i].Address);*

*sw.WriteLine(Providers.GetProviders[i].Phone.RawString());*

*sw.WriteLine(Providers.GetProviders[i].ContactFaceName);*

*sw.WriteLine();*

*}*

*for (int i = 0; i < Supplies.Count; i++)*

*{*

*sw.WriteLine("ПОСТАВКА");*

*sw.WriteLine(Supplies.GetSupplies[i].Item.Name);*

*sw.WriteLine(Supplies.GetSupplies[i].ItemCount.ToString());*

*sw.WriteLine(Supplies.GetSupplies[i].Provider.Name);*

*sw.WriteLine(Supplies.GetSupplies[i].Cost.ToString());*

*sw.WriteLine(Supplies.GetSupplies[i].Term);*

*sw.WriteLine();*

*}*

*sw.Close();*

*}*

*}*

*}*