Федеральное государственное бюджетное образовательное учреждение высшего образования

«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»

Департамент анализа данных,
принятия решений и финансовых технологий

Пояснительная записка к курсовой работе
(по дисциплине «Современные технологии программирования»)
на тему:

Информационная система «Учет товаров магазина аудиотехники»

Выполнил:
студент группы ПИ18-3 факультета
«Прикладная математика и
информационные технологии»
_____Комлев Н. П.
(Подпись)

Тел: 8 (915) 006-3271

Научный руководитель:
доцент, к.т.н. Горелов С. В.
_____
(Подпись)

2020 г.

Оглавление

Введение

Навыки в анализе предметной области и собственно разработка сложного программного Windows-приложения являются основными для студента направления «Прикладная информатика» и именно получение этих навыков и является главное целью текущей курсовой работы. Основной инструмент выполнения работы – объектно-ориентированный язык программирования C# и среда разработки Visual Studio. Сильная сторона языка C# - удобство и эффективность в разработке настольных предложений. Выполнение данной работы разовьет навыки разработки и анализа предметной области.

Для разработки информационной модели задачи и создания архитектуры приложения необходим анализ предметной области. Для качественного анализа необходимо выделить наиболее важные объекты и процессы автоматизации, а также разработку методов, IT-подходов и функций для решения проблем предметной области.

Помимо практических навыков в разработке, важным приобретением станет умение оформлять официальную документацию, в том числе пояснительную записку, отвечающую таким требованиям, как полнота, лаконичность, техническая корректность и т.д. Техническая документация является неотъемлемой частью программного продукта, является одним из важнейших посредников между разработчиком и конечным пользователем. Таким образом, документация определяет качество продукта. Поэтому умение грамотно оформить документацию так же важно для разработки программы, как и собственно разработка.

1. Постановка задачи

В соответствии с выбранной темой требуется разработать Windows-приложение, выполняющее учет товаров, прибывших на один из складов магазина аудиотехники, а также позволяющее формировать содержательные отчеты по наличию этих товаров.

Приложение должно содержать средства манипулирования объектами, сохранения их в таблицах базы данных, редактированию значений полей этих

таблиц, добавление и удаление этих объектов, с сохранением введенных изменений в базу.

Общие требования:
1. В курсовом проекте должна быть разработана информационная модель предметной области, представленная в виде пользовательских классов или таблиц БД.
2. Должно быть разработано несколько форм пользовательского интерфейса.
3. Разработчик самостоятельно определяет интерфейс программы и ее функциональность, однако для получения максимальной оценки приложение в обязательном порядке независимо от предметной области, указанной в задании, должно выполнять следующие операции:

- Отображать в сетках DataGridView данные предметной области:
- Для информационной модели, основанной на списках класса List<T>, на момент первого запуска программы допускается отсутствие файла. В этом случае списки объектов должны быть созданы в программе на этапе разработки.
- Для информационной модели, основанной на БД (Access, SQL Server), таблицы должны быть предварительно заполнены записями.
- Реализовать добавление в источник данных нового объекта, удаление объекта из источника данных, редактирование объекта источника данных.
- Реализовать фильтрацию записей источника данных, удовлетворяющих введенному пользователем сложному критерию.
- Реализовать сортировку записей источника данных, включая многоуровневую.
- Сохранять источник данных: для информационной модели, основанной на списках, в файле, используя XML-сериализацию (или Json). для информационной модели, основанной на таблицах БД, в базе данных.
- При запуске программы загрузить сохраненные данные из файла или базы данных.
- Используя меню или панель инструментов, вызвать приложение Блокнот для просмотра справки о программе: файл Help.txt текущего каталога программы.
- Создать пункт меню «Об авторе» с выводом соответствующей информации (MessageBox).
- Разработать несколько полезных пользователю функций для отображения статистических данных, например, средних, максимальных или минимальных значений, данных для построения гистограммы или графика и т.п.

4. Программа не должна завершаться аварийно: сообщения о некорректном вводе данных, противоречивых или недопустимых значениях данных, при отсутствии данных по функциональному запросу пользователя и других нештатных ситуациях отображать в окнах сообщений.

5. Программа должна быть читабельной и содержать полезные комментарии.
6. Интерфейс программы должен быть эргономичным и интуитивно понятным.
7. Использование самостоятельно изученных технологий (сохранение настроек
программы, интернационализация, вывод диаграмм и т.д.) будет вознаграждено дополнительными баллами.

2. Описание предметной области

В разрабатываемом приложении предметной областью автоматизации являются складские помещения магазина аудиотехники. Для магазина учет товаров является единственным способом организации деятельности по привозу и хранению товаров.

Деятельность склада организована следующим образом: сотрудник, отвечающий за учет (прием и регистрацию) товара по прибытии очередной партии обязан указать новоприбывший товар в базе данных. Для этого ему необходимо добавить записи с указанием соответствующих показателей (модель, категория, производитель, цена, время поступления, фактическое наличие, поставщик, склад хранения), а в случае отсутствия уже имеющихся показателей, добавить в базу и их.

Также для проверки и инвентаризации товаров на складе ответственному лицу необходимо осуществлять поиск товаров по базе и формировать отчетность по их наличию. В случае фактического отсутствия товара на складе, сотруднику следует внести соответствующие изменения в базу.

Для контроля за действиями сотрудников, отвечающих за учет товаров на складе, программа ведет журнал о любых изменениях в базе. В случае какой-либо ошибки и просчета, всегда можно определить, когда и кем была совершена оплошность.

Подводя учет и формируя отчетность, ответственные за это дело сотрудники улучшают работу магазина, снабжая его всей необходимой информацией о поставках и имеющемся товаре.

3. Актуальность автоматизации

Деятельность магазина аудиотехники напрямую связана с наличием собственно товара, который он намерен продать, чтобы получать доход от своего предприятия. Учет наличия этого товара на складах необходим для ведения бизнеса.

Сроки поступления товаров, их количество, их цены, информация о производителе и поставщиках – все эти данные несут в себе главный смысл учета и для того, чтобы эффективно это все отслеживать, необходимо обрабатывать очень большое количество информации.

В день могут привести сотню различных товаров, на складе могут храниться тысячи разных коробок, какой-то товар может пропасть и обо всем этом должна быть официальная документация. Если пытаться вручную вести учет по хранящимся бумагам о доставке и наличии товаров, работа ответственных за это дело сотрудников может стать более чем неэффективной. Поэтому автоматизация данной предметной области является актуальной задачей.

И программа АудиоСклад является эффективным решением для решения этой задачи. Она позволит управляться с объемной рутинной работой гораздо быстрее, исключая какие-либо неточности в ведении записей, позволяя с легкостью находить ошибки в работе персонала склада.

4. Описание программы

4.1 Алгоритмические решения

Все данные об объектах хранятся в базе данных, расположенной на запущенном на рабочей машине SQL-сервере. Разработка базы данных осуществлялась посредством программы SQL Server 2014 Management Studio, и ее структура будет рассмотрена в пункте 5.1 (рис. 5.1.1) данной пояснительной записки.

При загрузке приложения все данные считываются из базы с помощью SQL-запросов и записываются в списки классов, соответствующих объектам. Каждый экземпляр класса – одна запись в таблице базы данных (например, из таблицы Goods (товары) данные записываются в экземпляры класса Goods (товар), и этот экземпляр добавляется в список List<Goods>).

Для универсального доступа к данным из всех форм внутри программы создан публичный статический класс public static class Singleton. Обращение к публичным полям этого класса возможно из любой части программы. Также этот класс хранит все необходимые методы для работы с данными, такие, как «сохранение всех внесенных изменений» или «добавление новых объектов в базу».

Использование статического класса является весьма эффективным способом решения проблемы постоянной передачи нужных сведений по ссылкам из формы на форму. Имея этот класс, мы можем взаимодействовать с данными на любой форме, а использование методов для работы с данными внутри этого класса освобождает код каждой из форм от излишних нагромождений, делая его читаемым.

Переходя непосредственно к функциональной стороне алгоритма, следует отметить, что в программе проведена принципиальная грань между отображением данных и их хранением. Внесенные изменения не торопятся быть отраженными в базе данных. Добавление объекта можно увидеть в

размещенных на рабочей области таблицах, однако, пока изменения не были сохранены, в базе данных новая запись не появится.

В программе налажен механизм отмены и сохранения изменений. Если пользователь случайно удалил, например, товар, если он не сохранил изменения, он может их отменить.

Этот механизм реализован с помощью того же статического класса Singleton, который хранит в своих полях данные о добавленных, измененных и удаленных объектах. При сохранении изменений на основе хранящихся в этих полях данных будет составлен SQL-запрос, соответствующий операции (INSERT, UPDATE и DELETE). При отмене изменений, данные в основных списках вернутся к первоначальному виду, а списки, хранящие изменения, очистятся.

Еще одна принципиальная граница в функционале наблюдается в разделении управления данными и собственно учета. Для каждой из этих задач выделены отдельные окна, и по запуску программы это становится очевидно.

Что касается **учета**, то на отдельной форме для этого предоставлен комплекс элементов. Ключевой элемент – таблица DataGridView, содержащая перечень всех наличествующих на складах товаров. Таблица привязана к классу Goods (товары) через посредника BindingSource, отображает всю информацию о каждом экземпляре класса. Списком-источником же является, так называемый, «шоулист» - список товаров (List<Goods>), который находится также в статическом классе Singleton. Этот список содержит те товары, которые должны быть отображены непосредственно в данный момент времени.

На форме учета присутствуют элементы управления, позволяющие проводить фильтрацию данных по множественным критериям. При настройке критериев, «шоулист» будет заполнен товарами, которые соответствуют критериям, и отобразит в таблице именно то, что и подразумевал пользователь.

Статический класс Singleton включает в себя методы по проведению одноуровневой и многоуровневой сортировки данных в списке «шоулист». В методах, вызываемых событиями работы с таблицей DataGridView, также присутствуют алгоритмы по визуальному отображению происходящих действий при сортировке (подсветка сортируемых столбцов, отображение порядка сортировки). Согласованное взаимодействие этих программных методов предоставляет пользователю удобный интерфейс по работе с таблицей товаров.

Помимо просмотра товаров и значений их характеристик, сортировки и фильтрации данных в разделе учета можно также формировать отчетность. По нажатии соответствующей кнопки отобразится интерфейс для формирования документа, содержание которого пользователь может задать самостоятельно.

В окне формирования отчета присутствует список пунктов, которые должны в конечном счете быть внесены в отчет. Присутствуют некоторые обязательные пункты: склады, на которых проводится учет; категории

товаров, среди которых проводится учет; модели товаров; производители; поставщики; время поступления; диапазон цен. Эти пункты не могут быть удалены из перечня информации, которая отобразится в отчете.

Однако существуют статистические и справочные пункты, которые пользователь может выбрать из элемента управления ComboBox и по желанию внести в список учитываемых пунктов. Такие пункты отображают информацию по типу: количество учитываемых товаров, сумма их цен, их средняя цена, непосредственно список учитываемых товаром и т.п.

Сформированный отчет будет отображен в текстовом документе.

Переходя к разделу **управления данными**, можно отметить, что работа с данными происходит в рамках определенного склада. Прежде чем манипулировать данными, необходимо выбрать склад, с которым работает пользователь. Это исключает ошибки в размещении товаров. Выбор соответствующего склада реализован с использованием элемента ComboBox. Функция добавления нового склада для рядового пользователя закрыт. При необходимости добавления нового склада (по причине аренды нового помещения, например), необходимо обратиться к менеджеру, который будет волен внести изменения непосредственно в базу данных.

После выбора склада пользователю будет предоставлен функционал по поиску необходимых товаров с использованием фильтрации по идентификационному номеру и модели (от остальных критериев фильтрации было решено отказаться, поскольку с точки зрения логики, поиск по, например, дате поступления в разделе управления данными не имеет большого смысла, а поиск, например, по категориям можно провести с помощью сортировки), результаты поиска отражаются в отдельном неосновном DataGridView элементе, в основной же таблице представлен весь список товаров, находящихся на конкретном складе.

Источником данных основной таблицы снова является «шоулист» класса Singleton, а таблицы поиска – локальный «шоулист» (также List<Goods>), хранящийся только на данной форме.

Обе таблицы обладают свойствами одноуровневой сортировки, однако, по причине реализации двухуровневой сортировки через взаимодействие со списком «шоулист» класса Singleton, таблица поиска была лишена возможности производить двухуровневую сортировку. С учетом того, что в таблице поиска уже показаны отфильтрованные данные, было принято решение, что двухуровневая сортировка не будет составлять обязательную часть функционала. При желании заказчика, продукт будет доработан.

Основная и побочная таблицы синхронизированы, так что, если пользователь изменит товар, находящий в таблице поиска, его оригинальное отображение в основной таблице будет также изменено.

Это реализовано с помощью проверки выбранного элемента. Если ячейка выделена в неосновной таблице, выделение в основной очищается. После каждого действия пользователя происходит налаженное обновление

данных в таблице, так что отображенные в них объекты всегда будут иметь актуальный характер.

Процесс взаимодействия с данными происходит с использованием трех кнопок: добавить (добавить можно товар, категорию, модель, производителя, поставщика), изменить (только товар), удалить (только товар). По нажатии на кнопку добавления отображаются формы, содержащие необходимое количество полей для заполнения. Введенные данные, при подтверждении, будут внесены в новый экземпляр класса, который добавится в соответствующий список. Все изменения, добавления и удаления записываются в отдельные списки, это необходимо, чтобы стала возможным отмена всех проведенных действий.

Для запоминания подобных действий рассмотрено несколько вариантов:

1) Добавлен новый объект;
2) Изменен существующий объект;
3) Изменен новый объект;
4) Удален существующий неизмененный объект;
5) Удален существующий измененный объект;
6) Удален новый объект;

В зависимости от вида действия, списки заполняются соответствующим образом.

Отдельно хотелось упомянуть о возможности перехода с любой формы на различные этапы работы с приложением. Это реализовано с помощью MenuStrip, в котором есть пункты выбора окон, к которым пользователь хотел бы перейти.

Таким образом, программа реализует весь необходимый пользователю функционал для работы с товарами на складе, манипулирования данными и учета этих товаров.

4.2 Описание интерфейса программы

В приложении для реализации интерфейса используется 12 элементов управления и компонентов:

- DataGridView;
- Label;
- Button;
- ListBox;
- CheckBox;
- Panel;
- TextBox;
- ComboBox;
- DateTimePicker;
- MenuStrip;

- ToolTip;
- BindingSource;

Запуская Windows приложение, мы попадаем на главную форму (рис. 4.2.1). На ней отображены кнопки, предлагающие перейти к двум основным областям работы: учет и управление данными. На главной форме, как и на всех не диалоговых формах присутствует элемент MenuStrip, который содержит функционал по сохранению и отмене произведенных в течение работы изменений, закрытие приложения и отображения справки (рис. 4.2.11) и информации об авторе.

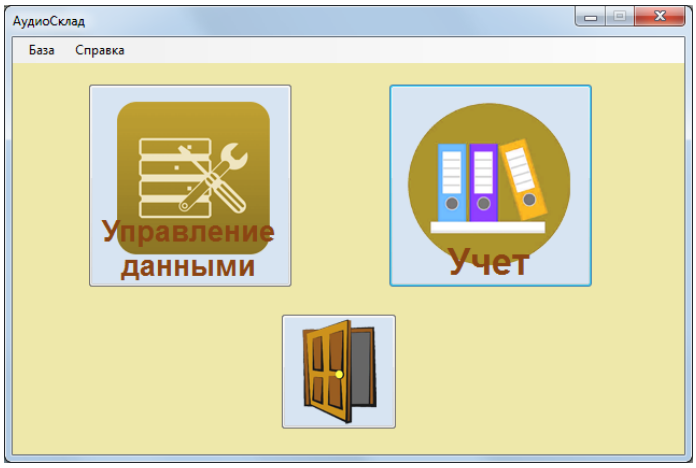


Рис. 4.2.1. Главное окно

Переходя в область учета, перед нами открывается форма «Учет» (рис. 4.2.2). На ней присутствуют элементы: ComboBox, TextBox, DateTimePicker, - которые способствуют проведению фильтрации. Заполняя поля корректными данными, после нажатия на кнопку поиск (лупа), в элементе DataGridView отобразятся товары, соответствующие выбранным критериям. Если данные введены некорректно, выдается ошибка.

Элемент MenuStrip – такой же, как и на главной форме, с добавлением пункта «Перейти», позволяющий осуществить переход на форму «Управление данным: Склады» (рис. 4.2.7).

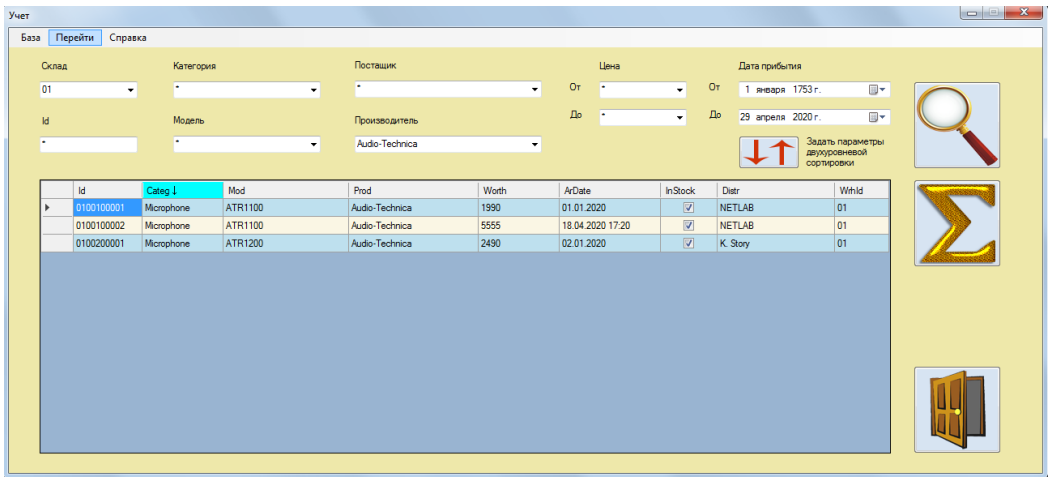
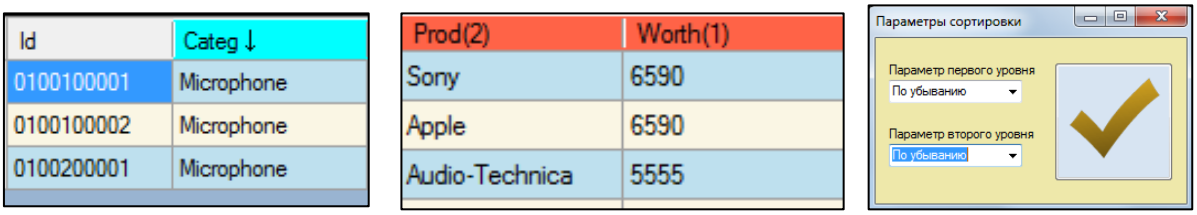


Рис. 4.2.2. Окно «Учет» с применением фильтрации

Сортировка данных осуществляется через нажатие на заголовок столбца. Простой одинарный клик производит одноуровневую сортировку (рис. 4.2.3 а), по нажатии на заголовок с зажатой клавишей Ctrl по очереди можно выбрать столбцы двухуровневой сортировки (рис. 4.2.3 б). Порядок же сортировки для каждого уровня выбирается в появляющемся окне выбора параметров сортировки, которое появляется при нажатии на кнопку «Задать параметры двухуровневой сортировки».



а) б) в)

Рис. 4.2.3. Сортировка одноуровневая (а), двухуровневая (б) и выбор параметров сортировки (в)

В области учета можно также формировать отчеты, которые предоставляют информацию по отраженным в данный момент товарам. В открывшемся окне «Подсчет» (рис. 4.2.4) можно увидеть элемент ListBox, содержащий пункты, которые в будущем будут показаны в отчете. Обязательные пункты обозначены звездочкой (*) и их нельзя исключить из списка. Добавить новые справочные и статистические пункты, выбранные в элементе ComboBox, можно по нажатии стрелочки. Выбранный пункт отразится в списке. Удалить пункт из списка можно, выбрав его в списке и нажав на крестик. Также можно очистить весь список необязательных пунктов, нажав на «корзину».

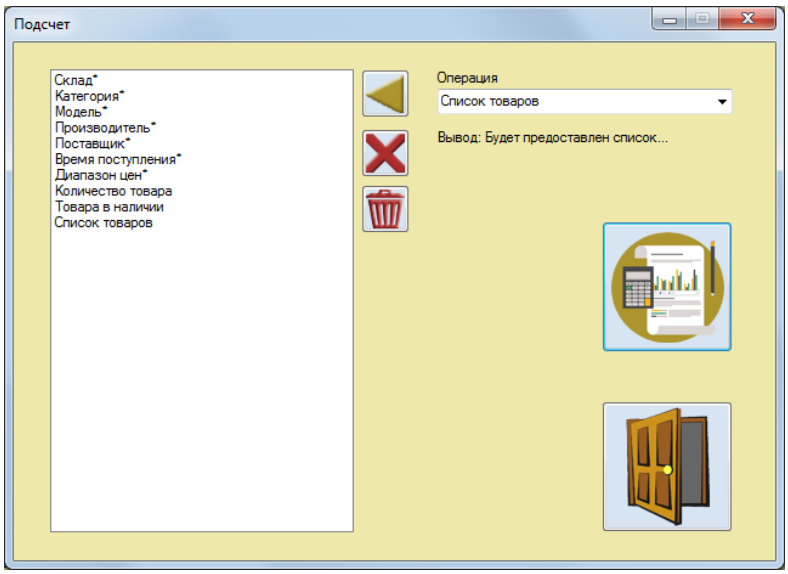


Рис. 4.2.4. Окно «Подсчет»

По нажатии на кнопку «Сформировать отчет» (рис. 4.2.5) откроется текстовый документ, который будет содержать текст будущего отчета. Для дальнейшей работы с отчетом или для лучшей презентации информации сформированный текст можно скопировать в текстовый редактор Word (рис. 4.2.6).

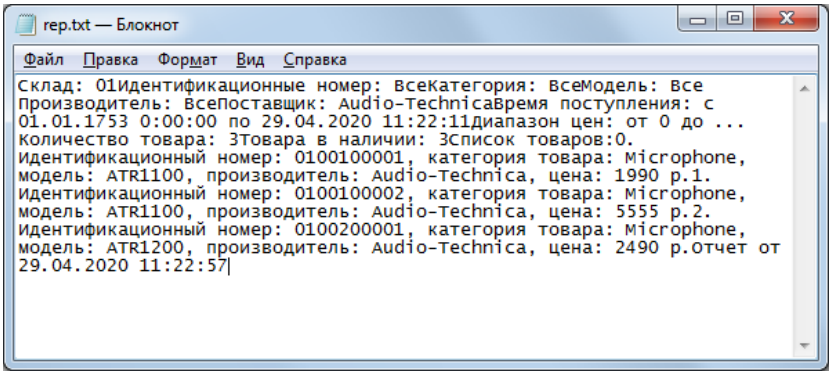


Рис. 4.2.5. Текстовый документ отчета в Блокноте

Склад: 01

Идентификационные номер: Все
Категория: Все
Модель: Все
Производитель: Все
Поставщик: Audio-Technica
Время поступления: с 01.01.1753 0:00:00 по 29.04.2020 11:15:31
Диапазон цен: от 0 до ...
Количество товара: 3
Товара в наличии: 3
Список товаров:
0. Идентификационный номер: 0100200001,
категория товара: Microphone,
модель: ATR1200,
производитель: Audio-Technica,
цена: 2490 р.

1. Идентификационный номер: 0100100002,
категория товара: Microphone,
модель: ATR1100,
производитель: Audio-Technica,
цена: 5555 р.

Рис. 4.2.6. Текстовый документ отчета в Word

С переходом в область управления данными открывается окно «Управление данными: Склады» (рис. 4.2.7). На данном этапе можно выбрать склад, в котором пользователь собирается работать. Также наличествует элемент MenuStrip с функционалом, как на форме «Учет».

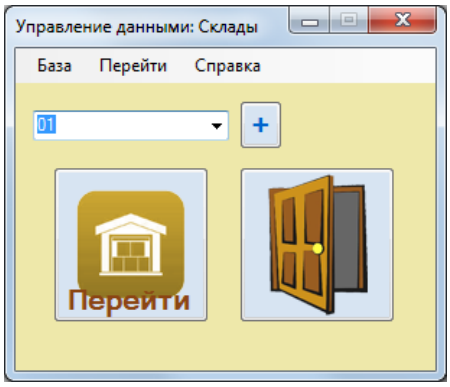


Рис. 4.2.7. Окно «Управление данными: Склады»

Выбрав склад, пользователь переходит к форме: «Управление данными: Товары» (рис. 4.2.8). В центре – таблица DataGridView, содержащая список всех товаров, зарегистрированных на выбранном складе. Она располагает теми же инструментами сортировки, как и таблица на форме «Учет» (для определения порядка двухуровневой сортировки также есть соответствующая кнопка). Для поиска конкретных товаров на форме присутствуют элементы для ввода идентификационного номера и модели, также кнопка для осуществления собственно поиска. Результаты отражаются во второй неосновной таблице DataGridView. Пользователь может взаимодействовать с

товарами (изменять, удалять) выбирая их как в основной таблице, так и в поисковой. Функционал таблиц синхронизирован.

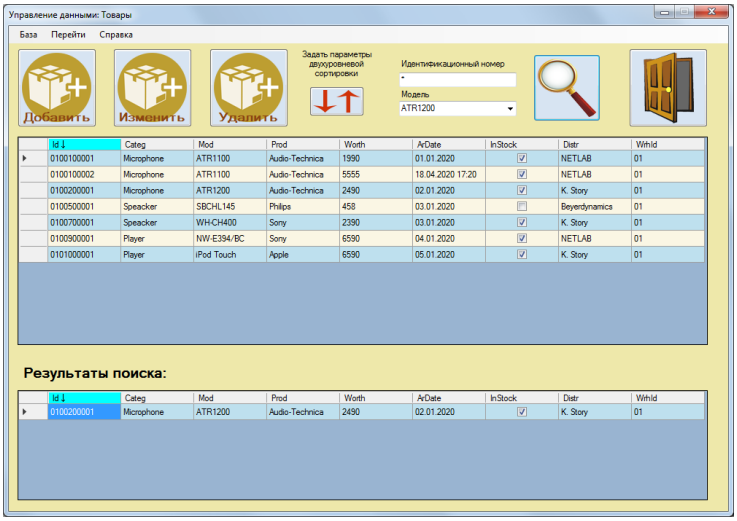
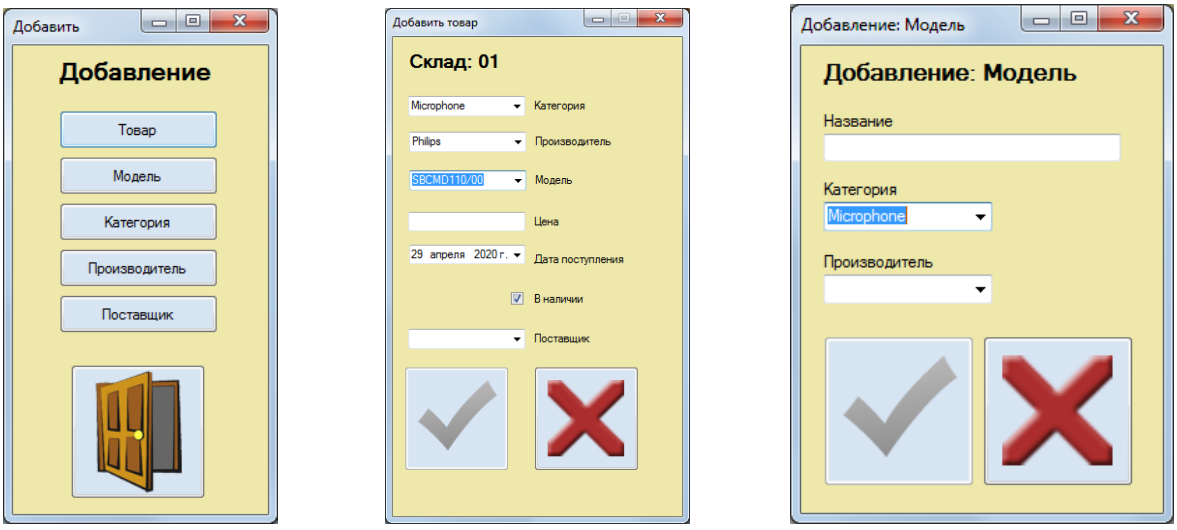
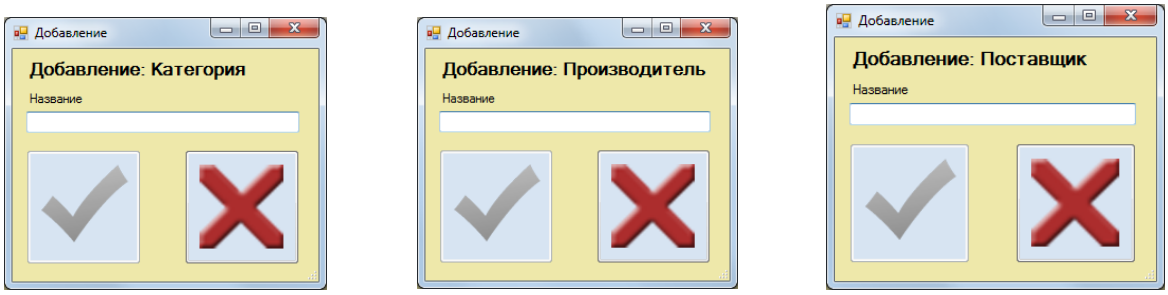


Рис. 4.2.8. Окно «Управление данными: Товары»

Добавление новых объектов в списки экземпляров классов происходит через отдельные формы. Нажав на кнопку «Добавить», будет предоставлен выбор объекта (рис. 4.2.9 а). В зависимости от выбранного объекта появляется соответствующее окно с необходимыми полями для заполнения. На каждой из окон налажена защита от пользовательских ошибок, реализованная через проверки и всплывающие окна сообщений.



а) б) в)



г) д) е)

Рис. 4.2.9. Окна добавления (а) товара (б), модели (в), категории (г), производителя (д) и поставщика (е)

Нажав на форме «Управление данными: Товары» кнопку «Изменить» пользователю предоставится новая форма (рис. 4.2.10), похожая на форму добавления, однако, уже содержащая данные по изменяемому товару.

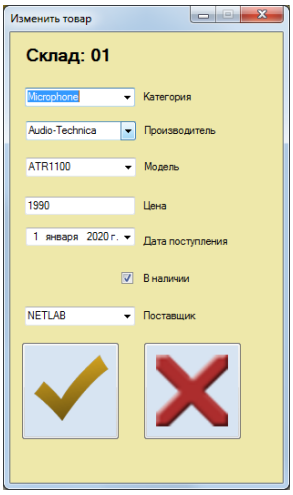


Рис. 4.2.10. Окно изменения товара

Для отображения внесенных изменений в базе необходимо на любой не диалоговой форме в элементе MenuStrip выбрать пункт «Сохранить изменения» (рис. 4.2.11). Для отмены изменений, соответственно, выбрать пункт «Отменить изменения».

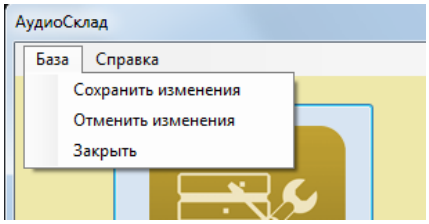


Рис. 4.2.11. Пункты MenuStrip для работы с базой и приложением

Также в MenuStrip в пункте «Справка» можно отобразить справку о программе (рис. 4.2.12). Справка откроется в текстовом документе Блокнота.

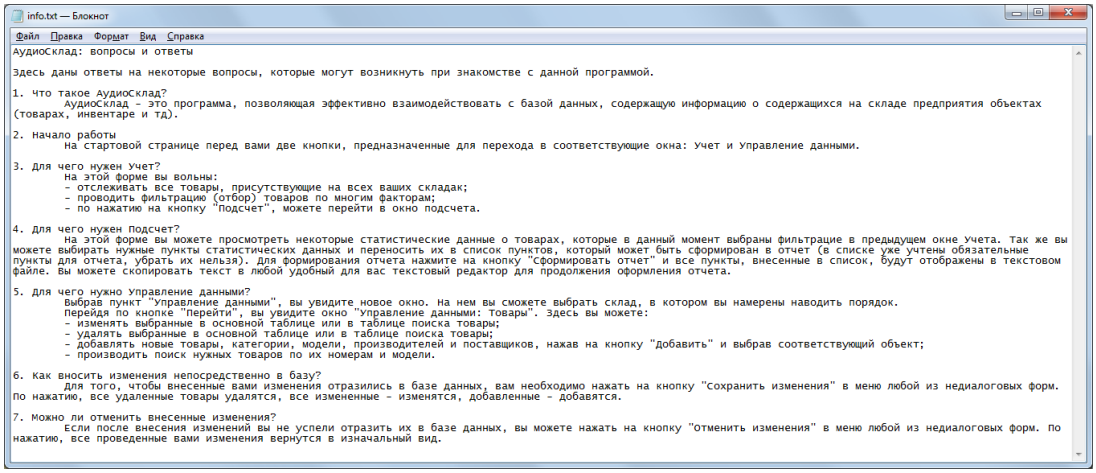


Рис.4.2.12. Текстовый документ со справкой

4.3 Состав приложения

В состав приложения входят следующие файлы и папки:

- KR.exe – программа;
- icons – папка с изображениями;
- info.txt – инструкция;
- rep.txt – отчеты;
- log.txt – журнал событий;
- Storage.bak – бэк-ап базы данных;

5. Назначение и состав классов программы

5.1 Диаграмма структуры классов

В разрабатываемой программе все данные хранятся в базе (рис. 5.1.1). Все созданные пользовательские классы соответствуют таблицам базы и имеют структуру, позволяющую отражать всю необходимую информацию.

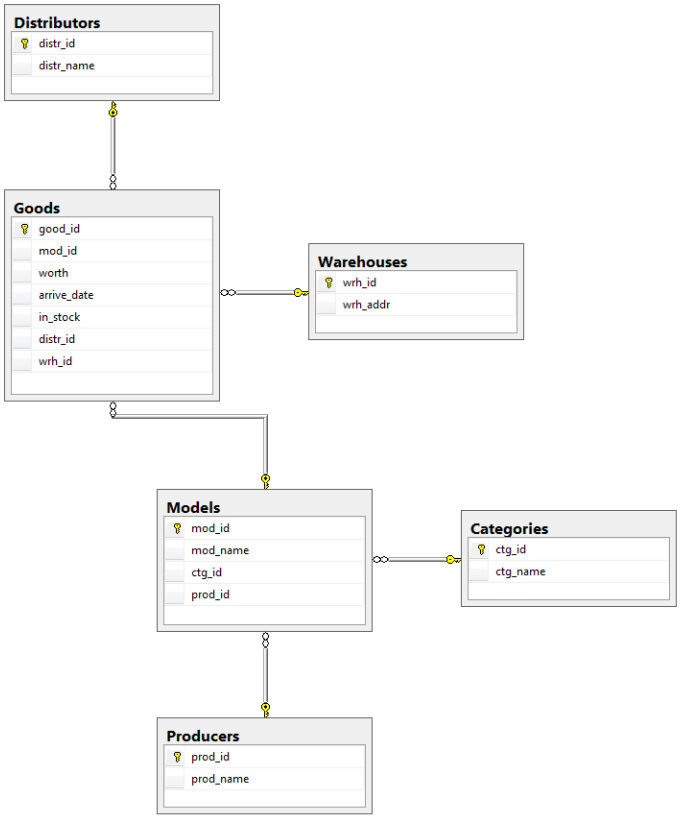


Рис. 5.1.1. Структура базы данных

5.2 Классы объектов учета

Для реализации поставленной задачи было разработано 6 одноименных классов: Warehouse (склад), Goods (товары), Category (категория), Model (модель), Producer (производитель), Distributor (поставщик).

Класс **Warehouse**. Содержит три свойства:

- Id - строковое значение, идентификационный номер склада.
- Address - строковое значение адреса склада.
- GoodsLst - список содержащихся на складе товаров.
- Warehouse(параметры) – параметрический конструктор, осуществляющий инициализацию свойств класса.
- ToString() – перезапись метода ToString() для приемлемого отображения экземпляра класса в полях ЭУ.

Класс **Goods** содержит восемь свойств:

- Id - строковое значение, идентификационный номер товара.
- Categ - экземпляр класса категории.
- Model - экземпляр класса модели.

- Producer - экземпляр класса производителя.
- Worth - вещественное значение цены.
- ArDate - дата поступления на склад.
- InStock - булевское обозначение: "в наличии".
- Distr - экземпляр класса поставщика.
- Goods(параметры) – параметрический конструктор, осуществляющий инициализацию свойств класса.
- ToString() – перезапись метода ToString() для приемлемого отображения экземпляра класса в полях ЭУ.

Класс **Category** содержит два свойства:

- Id - строковое значение, идентификационный номер категории товара.
- Name - строковое название категории.
- Category(параметры) – параметрический конструктор, осуществляющий инициализацию свойств класса.
- ToString() – перезапись метода ToString() для приемлемого отображения экземпляра класса в полях ЭУ.

Класс **Model** содержит четыре поля:

- Id - строковое значение, идентификационный номер модели.
- Name - строковое название модели.
- Ctg - экземпляр класса категории модели.
- Prod - экземпляр класса производителя.
- Model(параметры) – параметрический конструктор, осуществляющий инициализацию свойств класса.
- ToString() – перезапись метода ToString() для приемлемого отображения экземпляра класса в полях ЭУ.

Класс **Producer** содержит два свойства:

- Id - строковое значение, идентификационный номер производителя.
- Name - строковое название производителя.
- Producer(параметры) – параметрический конструктор, осуществляющий инициализацию свойств класса.
- ToString() – перезапись метода ToString() для приемлемого отображения экземпляра класса в полях ЭУ.

Класс **Distributor** содержит два свойства:

- Id - целочисленное значение, идентификационный номер поставщика.
- Name - строковое название компании поставщика.
- Distributor(параметры) – параметрический конструктор, осуществляющий инициализацию свойств класса.

- ToString() – перезапись метода ToString() для приемлемого отображения экземпляра класса в полях ЭУ.

5.3 Служебные классы

Центральной частью всей программы является публичный статический класс **Singleton**. В нем происходит взаимодействие с данными, связанными с базой данных. Ключевыми элементами являются публичные статические поля, являющиеся списками для хранения соответствующих объектов:

- public static List<Warehouse> wrhlist – список складов;
- public static List<Distributor> dislist – список поставщиков;
- public static List<Category> ctglist – список категорий;
- public static List<Producer> prolist – список производителей ;
- public static List<Model> modlist – список моделей;
- public static List<Goods> showlist – список товаров;

Поля, являющиеся переменными или словарем, хранящими старшие номера идентификационных номеров (запоминают старший и накапливаются при добавлении):

- public static int wr_id – переменная, содержащая старший номер склада;
- public static int dis_id – переменная, содержащая старший номер поставщика;
- public static int ctg_id – переменная, содержащая старший номер категории;
- public static int pro_id – переменная, содержащая старший номер производителя;
- public static int mod_id – переменная, содержащая старший номер модель;
- public static Dictionary<string, int> good_id – словарь, содержащий старший номер товара определенной модели (ключ – номер модели, значение – старший номер);

Поля, являющиеся списками для хранения соответствующих объектов:

- public static List<Goods> newgoodlist – список новых товаров;
- public static List<Goods> chgoodlist – список измененных товаров;
- public static List<Goods> delgoodlist – список удаленных товаров;
- public static List<Model> newmodlist– список новых моделей;
- public static List<Category> newctglist – список новых категорий;
- public static List<Producer> newprolist – список новых производителей;
- public static List<Distributor> newdislist – список новых поставщиков;

Также присутствует поле, хранящее путь до сервера для подключения к базе данных:

- public static string connectionString;

Описание методов класса Singleton:

- Singleton() – конструктор по умолчанию.
- RefreshShowList() – метод, предназначенный для обновления «шоулиста», для вывода всех товаров со всех складов
- TwoLevelSort(параметры) – метод, осуществляющий двухуровневую сортировку переданного списка товаров (однако результат будет записан в список «шоулист» класса Singleton) по переданным номерам столбцов и порядкам сортировки.
- OneLevelSort(параметры) – метод, осуществляющий одноуровневую сортировку переданного списка товаров по переданному номеру столбца и порядку сортировки.
- ChangeNew(параметры) – метод, осуществляющий действия по организации хранения данных при изменении нового товара. В качестве параметров передаются значения свойств измененного объекта (номер, модель и т.д.);
- ChangeGen(параметры) – метод, осуществляющий действия по организации хранения данных при изменении существующего товара. В качестве параметров передаются значения свойств измененного объекта (номер, модель и т.д.);
- DeleteNew(параметры) – метод, осуществляющий действия по организации хранения данных при удалении нового товара. В качестве параметров передаются номер товара и номер склада;
- DeleteGenNotChanged(параметры) – метод, осуществляющий действия по организации хранения данных при удалении существующего не измененного товара. В качестве параметров передаются номер товара и номер склада;
- DeleteGenChanged(параметры) – метод, осуществляющий действия по организации хранения данных при удалении существующего измененного товара. В качестве параметров передаются номер товара и номер склада;
- SaveChanges() – метод, осуществляющий действия по сохранению введенных изменений в базе данных.
- CancelChanges() – метод, осуществляющий действия по организации перераспределения данных для приведения их в изначальный вид при отмене изменений.
- Exit() – метод, осуществляющий закрытие приложения.
- Log(параметры) – метод, осуществляющий заполнение журнала log.txt сообщениями, переданными в качестве параметра.

Program – класс, содержащий метод Main, который позволяет запустить программу, открывая главную форму Form1.

5.4 Формы

Ниже представлены пользовательские классы, наследующие базовую функциональность от класса Form.

Form1 – класс, содержащий обработчики событий и методы по переходу на другие формы.

AccountF – класс, содержащий обработчики событий и методы для работы с отображением списка товаров в таблице, фильтрации данных, графического отображения действия по сортировке, а также перехода на другие формы.

MngWrh – класс, содержащий обработчики событий и методы для перехода на другие формы.

CountF – класс, содержащий обработчики событий и методы для работы с элементами управления в угоду формирования отчетности и перехода на другие формы.

MngGds – класс, содержащий обработчики событий и методы для работы с элементами управления в угоду отображения товаров, их поиска, фильтрации и перехода на другие формы.

ChAdd – класс, содержащий обработчики событий и методы для перехода на новые формы.

GdAdd – класс, содержащий обработчики событий и методы для работы с элементами управления в угоду добавления нового товара.

GdChan – класс, содержащий обработчики событий и методы для работы с элементами управления в угоду изменения товара.

ObjAdd – класс, содержащий обработчики событий и методы для работы с элементами управления в угоду добавления нового объекта.

ModAdd – класс, содержащий обработчики событий и методы для работы с элементами управления в угоду добавления новой модели.

SortF – класс, содержащий обработчики событий и методы для работы с элементами управления в определения порядков двухуровневой сортировки.

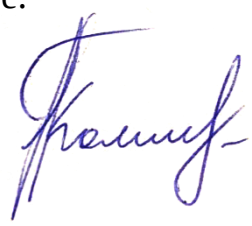
Заключение

Разработанное Windows приложение отвечает поставленным задачам и обозначенным требованиям. Справочно-информационная система, написанная на языке программирования C# с использованием таких инструментов, как методы, классы, свойства, поля и взаимодействием с сервером базы данных, осуществляет автоматизацию рабочего процесса предметной области. Реализация задач потребовала обращение к объектам операционной системы, таких как файловая система и процессы.

Программа обладает широким потенциалом для повышения эффективности функционала. Огромное желание вызывает организация взаимодействия с текстовым редактором Word при формировании отчетности, которое не было достигнуто по причине требований глубокого изучения документации соответствующих библиотек.

Список литературы

1. Горелов С.В., Волков А.Г. Разработка Windows-приложений. Часть 1. Учебное пособие. Образовательный портал Финансового университета. 2018.
2. Горелов С.В., Волков А.Г. Учебное пособие по дисциплине «Современные языки программирования». Разработка приложений на языке C# для работы с базами данных. – М.: Финансовый университет, 2016.
3. Горелов С.В. Разработка Windows-приложений. Часть 2. Учебное пособие. Образовательный портал Финансового университета. 2018.
4. Г. Шилдт. Полный справочник по C#. - М.: «Вильямс», 2004.
5. Официальный сайт Microsoft: [Интернет-ресурс]. URL: https://msdn.microsoft.com org.
6. 3. Проект с локальной базой данных SQL Server Express LocalDB [плейлист видеороликов лекций Горелова С.В.] // YouTube. https://www.youtube.com/playlist?list=PLcm83Nw-E0gAY0AQ3FMJxyAT12Ybr1b3-



Приложение. Исходный код программы

Файл Program.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace KR
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
```

```
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Файл Form1.cs:

```
using System;
using System.Data.SqlClient;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Collections;
using System.Net;

namespace KR
{
    public partial class Form1 : Form
    {
        public int transit = 0;
        //Домашняя форма, имеет возможность перехода на форму Учета товаров и на форму
Управления данными: Склад
        public Form1()
        {
            InitializeComponent();
        }

        public new void Show()
        {
            //Дополнение функции при появлении
            base.Show();
            //Проверка маршрута при отображении формы
            switch (transit)
            {
                //Переход в учет
                case 1:
                    DataRepr.PerformClick();
                    break;
                //Переход в склады
                case 2:
                    DataMng.PerformClick();
                    break;
                //Остаться
                default:
                    break;
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //Инициализируем класс Singleton
            Singleton.RefreshShowList();
        }

        private void DataRepr_Click(object sender, EventArgs e)
        {
            //Переход на форму учета
            Form newfrm = new AccountF(this);
```

```
            newfrm.Show();
            this.Hide();
        }

        private void DataMng_Click(object sender, EventArgs e)
        {
            //Переход на форму управления данными: склад
            Form newfrm = new MngWrh(this);
            newfrm.Show();
            this.Hide();
        }

        private void ExitBtn_Click(object sender, EventArgs e)
        {
            //Закрываем приложение
            Singleton.Exit();
        }

        private void SaveStrip_Click(object sender, EventArgs e)
        {
            //Сохраняем изменения
            Singleton.SaveChanges();
        }

        private void ExitStrip_Click(object sender, EventArgs e)
        {
            //Закрываем приложение
            Singleton.Exit();
        }

        private void CancelStrip_Click(object sender, EventArgs e)
        {
            //Отменяем изменения
            Singleton.CancelChanges();
        }

        private void ОбАвтореToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Программу разработал студент 2-го курса группы ПИ18-3 Комлев
Н.П.",
                "Версия 1.0", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void ОПрограммеToolStripMenuItem_Click(object sender, EventArgs e)
        {
            System.Diagnostics.Process.Start("notepad.exe", "info.txt");
        }

    }


}
```

Файл AccountF.cs:

```
using System;
using System.Data.SqlClient;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using System.Windows.Forms;
using System.Collections;
using System.Net;
using System.Windows.Input;

namespace KR
{
    public partial class AccountF : Form
    {
        //Форма Учета товара. Получает ссылку на домашнюю форму
        //Можно перейти на форму Подсчет, вернуться на домашнюю форму
        Form1 oldfrm;

        int transit = 0; //для определения окрываемой формы для перехода

        int sort = -1;
        int lastsortind = 0; //столбец для одноуровневой сортировки
        bool isonesort = true; //проверка на то, выделен ли столбец для одноуровневой
сортировки
        int firstlevelind; //первый столбец для двухуровнеой сортировки
        bool istwofirst = false; //проверка на то, выделен ли первый столбец для
двухуровневой сортировки
        int secondlevelind; //второй столбец для двухуровнеой сортировки
        bool istwosecond = false; //проверка на то, выделен ли второй столбец для
двухуровневой сортировки

        public string level1sort = "По убыванию"; //параметр первого уровня сортировки
        public string level2sort = "По убыванию"; //параметр первого уровня сортировки

        public AccountF(Form1 oldfrm_)
        {
            InitializeComponent();
            //Получаем ссылку на предыдущую форму
            oldfrm = oldfrm_;
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            // Устанавливаем "начальные значения" в комбобоксах
            CtgCmbBox.Items.Add("*");
            MdlCmbBox.Items.Add("*");
            PrdCmbBox.Items.Add("*");
            DisCmbBox.Items.Add("*");
            WthFromCmbBox.Items.Add("*");
            WthToCmbBox.Items.Add("*");
            WrhCmbBox.Items.Add("*");

            GGds.EnableHeadersVisualStyles = false;
            GGds.Columns[0].HeaderText += "↓";
            GGds.Columns[0].HeaderCell.Style.BackColor = Color.Aqua;

            // Заполняем комбобоксы для поиска
            for (int i = 0; i < Singleton.wrhlist.Count; i++)
            {
                WrhCmbBox.Items.Add(Singleton.wrhlist[i].ToString());
            }
            for (int i = 0; i < Singleton.ctglist.Count; i++)
            {
                CtgCmbBox.Items.Add(Singleton.ctglist[i].ToString());
            }
            for (int i = 0; i < Singleton.modlist.Count; i++)
            {
                MdlCmbBox.Items.Add(Singleton.modlist[i].ToString());
            }
            for (int i = 0; i < Singleton.prolist.Count; i++)
```

```
            {
                PrdCmbBox.Items.Add(Singleton.prolist[i].ToString());
            }
            for (int i = 0; i < Singleton.dislist.Count; i++)
            {
                DisCmbBox.Items.Add(Singleton.dislist[i].ToString());
            }
            // Заполняем шоулист  (спсиок товаров для вывода)
            Singleton.RefreshShowList();

            goodsBindingSource.DataSource = Singleton.showlist;
        }

        private void GGds_DataBindingComplete(object sender,
DataGridViewBindingCompleteEventArgs e)
        {
            // Выполняем "покрас" таблицы
            for (int i = 0; i < GGds.Rows.Count; i++)
            {
                if (i % 2 == 0)
                    GGds.Rows[i].DefaultCellStyle.BackColor = Color.FromArgb(190, 224,
238);
                else
                    GGds.Rows[i].DefaultCellStyle.BackColor = Color.FromArgb(250, 246,
228);

            }
        }

        private void SearchBtn_Click(object sender, EventArgs e)
        {
            try
            {
                //Заполняем переменные данными из элементов поиска
                if (IdTxtBox.Text.Length <= 0)
                    IdTxtBox.Text = "*";
                DateTime dtfsel = DatePickFrom.Value;
                DateTime dttsel = DatePickTo.Value;
                string idsel = IdTxtBox.Text;
                string wrhsel = WrhCmbBox.Text;
                string ctgsel = CtgCmbBox.Text;
                string mdlsel = MdlCmbBox.Text;
                string prdsel = PrdCmbBox.Text;
                string dissel = DisCmbBox.Text;
                double wthfsel;
                double wthtsel;

                if (WthFromCmbBox.Text is "*") wthfsel = 0;
                else wthfsel = Convert.ToDouble(WthFromCmbBox.Text);

                if (WthToCmbBox.Text is "*") wthtsel = 999999999;
                else wthtsel = Convert.ToDouble(WthToCmbBox.Text);

                //Передаем данные в функцию поиска
                Searching(idsel, wrhsel, ctgsel, mdlsel, prdsel, dissel, wthfsel,
wthtsel, dtfsel, dttsel);
            }

            catch
            {
                //В случае некорректно введенных данных
                MessageBox.Show(
                    "Некорректный ввод",
                    "Сообщение",
                    MessageBoxButtons.OK,
```

```
                    MessageBoxIcon.Information,
                    MessageBoxDefaultButton.Button1);
            }
        }

        private void GGds_ColumnHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
        {

            if (Control.ModifierKeys == Keys.Control)
            {
                // Многоуровневая сортировка по нажатию на имя столбца с зажатой клавишей
Ctrl
                if (isonesort)
                {
                    //Если до этого была выбрана одноуровневая сорторовка
                    ChangeSelection(lastsortind, 1);
                    isonesort = false;

                    firstlevelind = e.ColumnIndex;
                    GGds.Columns[firstlevelind].HeaderCell.Style.BackColor =
Color.Tomato;
                    GGds.Columns[firstlevelind].HeaderText += "(1)";
                    istwofirst = true;
                    goodsBindingSource.ResetBindings(false);
                }
                else if (!istwosecond & e.ColumnIndex != firstlevelind)
                {
                    //Если до этого был выбран только первый столбец для двухуровневой
сортировки
                    secondlevelind = e.ColumnIndex;
                    GGds.Columns[secondlevelind].HeaderCell.Style.BackColor =
Color.Tomato;
                    GGds.Columns[secondlevelind].HeaderText += "(2)";
                    istwosecond = true;
                    Singleton.TwoLevelSort(firstlevelind, secondlevelind, level1sort,
level2sort, Singleton.showlist);
                    goodsBindingSource.DataSource = Singleton.showlist;
                }
                else
                {
                    //Если до этого был выбрана двухуровневая сортировка
                    ChangeSelection(firstlevelind, 3);
                    if (istwosecond)
                    {
                        ChangeSelection(secondlevelind, 3);
                        istwosecond = false;
                    }

                    firstlevelind = e.ColumnIndex;
                    GGds.Columns[firstlevelind].HeaderCell.Style.BackColor =
Color.Tomato;
                    GGds.Columns[firstlevelind].HeaderText += "(1)";
                    istwofirst = true;
                    goodsBindingSource.ResetBindings(false);
                }

            }
            else
            {
                // Одноуровневая сортировка по нажатию на имя столбца
                if (isonesort)
                {
                    //Если до этого была выбрана одноуровневая сортировка
                    ChangeSelection(lastsortind, 1);
```

```
                    isonesort = false;
                }
                else
                {
                    //Если до этого была выбрана двухуровневая сортировка
                    ChangeSelection(firstlevelind, 3);
                    istwofirst = false;
                    if (istwosecond)
                    {
                        ChangeSelection(secondlevelind, 3);
                        istwosecond = false;
                    }

                }
                sort *= -1;
                if (sort > 0)
                    GGds.Columns[e.ColumnIndex].HeaderText += "↑";
                else
                    GGds.Columns[e.ColumnIndex].HeaderText += "↓";
                //Собственно сортировка
                Singleton.OneLevelSort(GGds.Columns[e.ColumnIndex].Index, sort,
Singleton.showlist);

                lastsortind = e.ColumnIndex;
                GGds.Columns[lastsortind].HeaderCell.Style.BackColor = Color.Aqua;
                goodsBindingSource.ResetBindings(false);
                isonesort = true;
            }
        }

        private void ChangeSelection(int i, int z)
        {
            //Функция для очистки предыдущего выделения столбцов
            var x = GGds.Columns[i].HeaderText;
            GGds.Columns[i].HeaderText = x.Substring(0, x.Length - z);
            GGds.Columns[i].HeaderCell.Style.BackColor = DefaultBackColor;
        }

        public void Searching(string idsel, string wrhsel, string ctgsel, string mdlsel,
string prdsel,
                              string dissel, double wthfsel, double wthtsel, DateTime
dtfsel, DateTime dttsel)
        {
            //Функция поиска данных
            //принимает на вход информацию, взятую из элементов поиска:
            //идентификационный номер, номер склада, номер категории, номер модели, номер
производителя,
            //номер поставщика, левая граница цены, правая граница цены,
            //левая граница времени прибытия, правая граница времени прибытия

            // Поиск данных по данным фильтров и заполнения шоулиста
            List<Goods> searchlist = new List<Goods>();
            Goods x;

            if (wrhsel == "*")
            {
                // Прогон всех складов и всех товаров
                for (int i = 0; i < Singleton.wrhlist.Count; i++)
                {
                    for (int j = 0; j < Singleton.wrhlist[i].GoodsLst.Count; j++)
                    {
                        x = Singleton.wrhlist[i].GoodsLst[j];

                        // Проверка всех полей на данные из поиска
                        if (((idsel is "*") | (idsel == x.Id)) &
```

```
                            ((ctgsel is "*") | (ctgsel == x.Categ.ToString())) &
                            ((mdlsel is "*") | (mdlsel == x.Mod.ToString())) &
                            ((prdsel is "*") | (prdsel == x.Prod.ToString())) &
                            ((dissel is "*") | (dissel == x.Distr.ToString())) &
                            ((x.Worth <= wthtsel) & (x.Worth >= wthfsel)) &
                            ((x.ArDate <= dttsel) & (x.ArDate >= dtfsel))
                            ) searchlist.Add(Singleton.wrhlist[i].GoodsLst[j]);
                    }
                }
            }
            else
            {
                // Прогон всех товаров определенного склада
                for (int j = 0; j < Singleton.wrhlist.Find(s => s.ToString() ==
wrhsel).GoodsLst.Count; j++)
                {
                    x = Singleton.wrhlist.Find(s => s.ToString() == wrhsel).GoodsLst[j];

                    // Проверка всех полей на данные из поиска
                    if (((idsel is "*") | (idsel == x.Id)) &
                        ((ctgsel is "*") | (ctgsel == x.Categ.ToString())) &
                        ((mdlsel is "*") | (mdlsel == x.Mod.ToString())) &
                        ((prdsel is "*") | (prdsel == x.Prod.ToString())) &
                        ((dissel is "*") | (dissel == x.Distr.ToString())) &
                        ((x.Worth <= wthtsel) & (x.Worth >= wthfsel)) &
                        ((x.ArDate <= dttsel) & (x.ArDate >= dtfsel))) searchlist.Add(x);
                }

            }
            Singleton.showlist = searchlist;

            goodsBindingSource.DataSource = Singleton.showlist;
        }

        private void CountBtn_Click(object sender, EventArgs e)
        {
            //Переход на форму подсчета
            Form newfrm = new CountF(WrhCmbBox.Text, CtgCmbBox.Text, IdTxtBox.Text,
MdlCmbBox.Text, DisCmbBox.Text,
                                     PrdCmbBox.Text, WthFromCmbBox.Text, WthToCmbBox.Text,
DatePickFrom.Value.ToString(),
                                     DatePickTo.Value.ToString());
            newfrm.ShowDialog();
        }

        private void SaveStrip_Click(object sender, EventArgs e)
        {
            //Сохраняем изменения
            Singleton.SaveChanges();
        }

        private void ExitStrip_Click(object sender, EventArgs e)
        {
            //Закрываем приложение
            Singleton.Exit();
        }

        private void CancelStrip_Click(object sender, EventArgs e)
        {
            //Отменяем изменения
            Singleton.CancelChanges();
            Singleton.RefreshShowList();
            SearchBtn.PerformClick();
        }
```

```
        private void ExitBtn_Click(object sender, EventArgs e)
        {
            //Указываем маршрут к главной
            transit = 0;
            //Закрываем форму
            this.Close();
        }

        private void Form2_FormClosing(object sender, FormClosingEventArgs e)
        {
            //Указываем путь
            oldfrm.transit = transit;
            //Возвращаемся на форму назад
            oldfrm.Show();
        }

        private void ОбАвтореToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Программу разработал студент 2-го курса группы ПИ18-3 Комлев
Н.П.",
                "Версия 1.0", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void ОПрограммеToolStripMenuItem_Click(object sender, EventArgs e)
        {
            System.Diagnostics.Process.Start("notepad.exe", "info.txt");
        }

        private void SortBtn_Click(object sender, EventArgs e)
        {
            //Переход к форме настройки двухуровневой сортировки
            Form newfrm = new SortF(this, null);
            newfrm.ShowDialog();
        }

        private void ГлавнаяToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //Указываем маршрут к главной
            transit = 0;
            //Закрываем форму
            this.Close();
        }

        private void УправлениеСкладыToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //Указываем маршрут к складам
            transit = 2;
            //Закрываем форму
            this.Close();
        }
    }
}
```

Файл MngWrh.cs:

```
using System;
using System.Data.SqlClient;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using System.Windows.Forms;
using System.Collections;
using System.Net;

namespace KR
{
    public partial class MngWrh : Form
    {
        //Форма Управление данными: Склад. Выбирается склад для работы.
        //Можно перейти на форму Управления данными: Товары, вернуться на домашнюю форму.
        public Form1 oldfrm;
        public int transit = 3; //для определения окрываемой формы для перехода

        public MngWrh(Form1 oldfrm_)
        {
            InitializeComponent();
            //Получаем ссылку на предыдущую форму
            oldfrm = oldfrm_;
        }

        private void Form3_Load(object sender, EventArgs e)
        {
            transit = 3;
            //Заоплняем комбобоксы названиями складов
            for (int i = 0; i < Singleton.wrhlist.Count; i++)
            {
                WrhCmbBox.Items.Add(Singleton.wrhlist[i].ToString());
            }

            WrhCmbBox.Text = WrhCmbBox.Items[0].ToString();
        }

        public new void Show()
        {
            //Дополнение функции при появлении
            base.Show();
            //Проверка маршрута при отображении формы
            switch (transit)
            {
                //Переход на главную
                case 0:
                    ExitBtn.PerformClick();
                    break;
                //Переход в учет
                case 1:
                    учетToolStripMenuItem.PerformClick();
                    break;
                //Остаться
                default:
                    break;
            }
        }

        private void WrhSelBtn_Click(object sender, EventArgs e)
        {
            //Переход к форме управления данными: Товары
            Form newfrm = new MngGds(this, WrhCmbBox.Text.ToString());
            newfrm.Show();
            this.Hide();
        }

        private void WrhAddBtn_Click(object sender, EventArgs e)
        {
            //Попытка добавления новых складов заблокирована
            MessageBox.Show(
```

```
                "Не хватает прав, обратитесь к администратору",
                "Сообщение",
                MessageBoxButtons.OK,
                MessageBoxIcon.Information,
                MessageBoxDefaultButton.Button1);
        }

        private void SaveStrip_Click(object sender, EventArgs e)
        {
            //Сохраняем изменения
            Singleton.SaveChanges();
        }

        private void ExitStrip_Click(object sender, EventArgs e)
        {
            //Закрываем приложение
            Singleton.Exit();
        }

        private void CancelStrip_Click(object sender, EventArgs e)
        {
            //Отменяем изменения
            Singleton.CancelChanges();
        }

        private void Form3_FormClosing(object sender, FormClosingEventArgs e)
        {
            //Передаем координаты перхода
            oldfrm.transit = transit;
            //Возвращаемся на форму назад
            oldfrm.Show();

        }

        private void ExitBtn_Click(object sender, EventArgs e)
        {
            //Сообщаем о переходе на главную
            transit = 0;
            //Закрываем форму
            this.Close();
        }

        private void ОбАвтореToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Программу разработал студент 2-го курса группы ПИ18-3 Комлев
Н.П.",
                "Версия 1.0", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void ОПрограммеToolStripMenuItem_Click(object sender, EventArgs e)
        {
            System.Diagnostics.Process.Start("notepad.exe", "info.txt");
        }

        private void УчетToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //Сообщаем о переходе в учет
            transit = 1;
            //Закрываем форму
            this.Close();
        }

        private void ГлавнаяToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //Сообщаем о переходе на главную
```

```
            transit = 0;
            //Закрываем форму
            this.Close();
        }

        private void WrhCmbBox_TextChanged(object sender, EventArgs e)
        {
            //Защита от пользовательского ввода
            if (!WrhCmbBox.Items.Contains(WrhCmbBox.Text))
                WrhCmbBox.Text = WrhCmbBox.Items[0].ToString();
        }
    }
}
```

Файл CountF.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Word = Microsoft.Office.Interop.Word;
using System.Reflection;
using System.IO;
using NPOI.XWPF.UserModel;


namespace KR
{
    public partial class CountF : Form
    {
        //Форма Подсчет
        //Можно перейти к формированию отчета, перейти назад на форму Учета
        bool nochanges = true;
        List<string> defaultlst = new List<string>(); //Список с обязательными пунктами
составления отчета
        string wrh, ctg, id, mod, dis, pro, wthfr, wthto, datfr, datto;

        public CountF(string wrh_, string ctg_, string id_, string mod_, string dis_,
string pro_, string wthfr_, string wthto_, string datfr_, string datto_)
        {
            //Передаем информацию с предыдущей формы о заданных фильтрах
            InitializeComponent();
            wrh = wrh_ == "*" ? "Все" : wrh_;
            ctg = ctg_ == "*" ? "Все" : ctg_;
            id = id_ == "*" ? "Все" : id_;
            mod = mod_ == "*" ? "Все" : mod_;
            dis = dis_ == "*" ? "Все" : dis_;
            pro = pro_ == "*" ? "Все" : pro_;
            wthfr = wthfr_ == "*" ? "0" : wthfr_;
            wthto = wthto_ == "*" ? "..." : wthto_;
            datfr = datfr_;
            datto = datto_;
        }

        private void Form4_Load(object sender, EventArgs e)
        {
            //Выводим начальную информацию
            if (InfCmbBox.Text == "Количество товара")
                InfLbl.Text += Singleton.showlist.Count.ToString();
            //Заполняем список обязательными пунктами
```

```
            for (int i = 0; i < ListBox.Items.Count; i++)
            {
                defaultlst.Add(ListBox.Items[i].ToString());
            }

        }

        private void InfCmbBox_TextChanged(object sender, EventArgs e)
        {
            //При выборе определенного пункта выводится соответствующая информация
            if (!InfCmbBox.Items.Contains(InfCmbBox.Text))
            {
                InfCmbBox.Text = "Количество товара";
                Msg1();
            }
            else
            {
                InfLbl.Text = "Вывод: ";
                InfLbl.Text += TextSwitch(InfCmbBox.Text);
            }
        }

        public string TextSwitch(string inc)
        {
            //Функция формирования ответа по соответствующему пункту
            //принимает текстовое значение пункта
            string res = "";
            switch (inc)
            {
                case "Количество товара":
                    res += Singleton.showlist.Count.ToString();
                    break;
                case "Средняя цена":
                    res += Math.Round(Singleton.showlist.Average(s => s.Worth), 2);
                    break;
                case "Сумма цен":
                    res += Singleton.showlist.Sum(s => s.Worth);
                    break;
                case "Товара в наличии":
                    res += Singleton.showlist.Count(s => s.InStock == true);
                    break;
                case "Самый дорогостоящий товар":
                    res += Singleton.showlist.Find(s => s.Worth ==
Singleton.showlist.Max(u => u.Worth)).ToString();
                    break;
                case "Самый дешевый товар":
                    res += Singleton.showlist.Find(s => s.Worth ==
Singleton.showlist.Min(u => u.Worth)).ToString();
                    break;
                case "Список товаров":
                    res += "Будет предоставлен список...";
                    break;
            }
            return res;
        }

        private void AddInfBtn_Click(object sender, EventArgs e)
        {
            //По нажатию кнопки, пункт расчета отобразится в списке
            nochanges = false;
            if (!ListBox.Items.Contains(InfCmbBox.Text))
                ListBox.Items.Add(InfCmbBox.Text);
            else Msg2();
        }
```

```
        private void ClearBtn_Click(object sender, EventArgs e)
        {
            //Удаляет указанный пункт расчета
            try
            {
                if (!defaultlst.Contains(ListBox.SelectedItem.ToString()))
                    ListBox.Items.RemoveAt(ListBox.SelectedIndex);
                else Msg3();
            }
            catch
            {
                Msg3();
            }
        }

        private void BinBtn_Click(object sender, EventArgs e)
        {
            //Очищает список от необязательных пунктов
            int i = 0;
            while (i < ListBox.Items.Count)
            {
                if (!defaultlst.Contains(ListBox.Items[i]))
                {
                    ListBox.Items.RemoveAt(i);
                    i -= 1;
                }
                i += 1;
            }
        }

        private void FormRepBtn_Click(object sender, EventArgs e)
        {
            //Функция для формирования отчета
            string report = String.Format("Склад: {0}\n\n\nИдентификационные номер:
{1}\nКатегория: {2}\nМодель: {3}\nПроизводитель: {4}\nПоставщик: {5}\n" +
                          "Время поступления: с {6} по {7}\nДиапазон цен: от {8} до {9}",
wrh, ctg, id, mod, dis, pro, datfr, datto, wthfr, wthto);
            for (int i = 0; i < ListBox.Items.Count; i++)
            {
                if (!defaultlst.Contains(ListBox.Items[i].ToString()) &
ListBox.Items[i].ToString() != "Список товаров")
                {
                    report += String.Format("\n{0}: {1}", ListBox.Items[i].ToString(),
TextSwitch(ListBox.Items[i].ToString()));
                }
                else if (ListBox.Items[i].ToString() == "Список товаров")
                {
                    report += "\nСписок товаров:";
                    for (int j = 0; j < Singleton.showlist.Count; j++)
                    {
                        report += String.Format("\n{0}. {1}", j,
Singleton.showlist[j].ToString());
                    }
                }
            }
            report += String.Format("\n\n\nОтчет от {0}", DateTime.Now);

            File.Delete("rep.txt");
            using (FileStream fs = new FileStream("rep.txt", FileMode.OpenOrCreate))
            {
                byte[] x = Encoding.Default.GetBytes(report);
                fs.Write(x, 0, x.Length);
            }
            System.Diagnostics.Process.Start("notepad.exe", "rep.txt");
        }
```

```
        public void Msg1()
        {
            MessageBox.Show(
                    "Некорректный ввод",
                    "Сообщение",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information,
                    MessageBoxDefaultButton.Button1);
        }

        public void Msg2()
        {
            MessageBox.Show(
                    "Данная операция уже имеется в списке",
                    "Сообщение",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information,
                    MessageBoxDefaultButton.Button1);
        }

        public void Msg3()
        {
            MessageBox.Show(
                    "Следует выбрать необязательный элемент отчета",
                    "Сообщение",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information,
                    MessageBoxDefaultButton.Button1);
        }

        private void ExitBtn_Click(object sender, EventArgs e)
        {
            //Закрываем форму
            DialogResult = DialogResult.OK;
        }

        private void Form4_FormClosing(object sender, FormClosingEventArgs e)
        {
            //Убеждаемся, хотим ли закрыть форму
            if (!nochanges)
            {
                DialogResult result = MessageBox.Show(
                    "Есть несохраненные данные. Закрыть?",
                    "Сообщение",
                    MessageBoxButtons.YesNo,
                    MessageBoxIcon.Information,
                    MessageBoxDefaultButton.Button1);

                if (result != DialogResult.Yes)
                    e.Cancel = true;
            }
        }
    }
}
```

Файл MngGds.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using System.Windows.Forms;

namespace KR
{
    public partial class MngGds : Form
    {
        //Форма Управление данными: Товары
        //Можно перейти к форме добавления объектов, форме изменения товара, возвратиться
на форму Управление данными: Склады
        readonly MngWrh oldfrm;

        List<Goods> showsearchlist = new List<Goods>();
        string wrh;
        int sort1 = -1;
        int sort2 = -1;

        int transit = 3;

        int lastsortind1 = 0; //столбец для одноуровневой сортировки первой таблицы
        bool isonesort1 = true; //проверка на то, выделен ли столбец для одноуровневой
сортировки первой таблицы
        int firstlevelind1; //первый столбец для двухуровнеой сортировки первой таблицы
        bool istwofirst1 = false; //проверка на то, выделен ли первый столбец для
двухуровневой сортировки первой таблицы
        int secondlevelind1; //второй столбец для двухуровнеой сортировки первой таблицы
        bool istwosecond1 = false; //проверка на то, выделен ли второй столбец для
двухуровневой сортировки первой таблицы

        public string level1sort = "По убыванию"; //параметр первого уровня сортировки
        public string level2sort = "По убыванию"; //параметр первого уровня сортировки

        int lastsortind2 = 0; //столбец для одноуровневой сортировки второй таблицы


        public MngGds(MngWrh oldfrm_, string wrh_)
        {
            InitializeComponent();
            //Получаем ссылку на предыдущую форму
            oldfrm = oldfrm_;
            //Получаем информацию о том, с каким складом работаем
            wrh = wrh_;
        }

        private void Form5_Load(object sender, EventArgs e)
        {
            //Заполняем таблицу
            RefreshShowList();
            goodsBindingSource.DataSource = Singleton.showlist;
            //Заполняем комбобокс для поиска
            ModCmbBx.Items.Add("*");
            for (int i = 0; i < Singleton.modlist.Count; i++)
            {
                ModCmbBx.Items.Add(Singleton.modlist[i].ToString());
            }
            dataGridView1.EnableHeadersVisualStyles = false;
            dataGridView2.EnableHeadersVisualStyles = false;
            dataGridView1.Columns[0].HeaderText += "↓";
            dataGridView1.Columns[0].HeaderCell.Style.BackColor = Color.Aqua;
            dataGridView2.Columns[0].HeaderText += "↓";
            dataGridView2.Columns[0].HeaderCell.Style.BackColor = Color.Aqua;

        }

        public void RefreshShowList()
```

```
        {
            //Функция для отображения всех товаров из данного склада
            var x = Singleton.wrhlist.Find(s => s.ToString() == wrh);
            Singleton.showlist.Clear();
            for (int i = 0; i < x.GoodsLst.Count; i++)
            {
                Singleton.showlist.Add(x.GoodsLst[i]);
            }
        }

        private void AddGoodBtn_Click(object sender, EventArgs e)
        {
            //Действия по нажатию на "Добавить", переход на новую форму
            Form newfrm = new ChAdd(this, wrh);
            newfrm.ShowDialog();
            RefreshShowList();
            goodsBindingSource.ResetBindings(false);


            ModCmbBx.Items.Clear();
            ModCmbBx.Items.Add("*");
            ModCmbBx.Text = "*";
            for (int i = 0; i < Singleton.modlist.Count; i++)
            {
                ModCmbBx.Items.Add(Singleton.modlist[i].ToString());
            }

            if (showsearchlist.Count > 0)
                SearchBtn.PerformClick();
        }

        private void SaveStrip_Click(object sender, EventArgs e)
        {
            //Сохраняем изменения
            Singleton.SaveChanges();
        }

        private void ExitStrip_Click(object sender, EventArgs e)
        {
            //Закрываем приложение
            Singleton.Exit();
        }

        private void CancelStrip_Click(object sender, EventArgs e)
        {
            //Отменяем изменения
            Singleton.CancelChanges();
            RefreshShowList();
            goodsBindingSource.ResetBindings(false);
            if (showsearchlist.Count > 0)
                SearchBtn.PerformClick();
        }

        private void DataGridView1_DataBindingComplete(object sender,
DataGridViewBindingCompleteEventArgs e)
        {
            // Выполняем "покрас" таблицы
            for (int i = 0; i < dataGridView1.Rows.Count; i++)
            {
                if (i % 2 == 0)
                    dataGridView1.Rows[i].DefaultCellStyle.BackColor =
Color.FromArgb(190, 224, 238);
                else
                    dataGridView1.Rows[i].DefaultCellStyle.BackColor =
Color.FromArgb(250, 246, 228);
```

```
            }
        }

        private void DataGridView1_ColumnHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            if (Control.ModifierKeys == Keys.Control)
            {
                // Многоуровневая сортировка по нажатию на имя столбца с зажатой клавишей
Ctrl
                if (isonesort1)
                {
                    //Если до этого была выбрана одноуровневая сорторовка
                    ChangeSelection1(lastsortind1, 1);
                    isonesort1 = false;

                    firstlevelind1 = e.ColumnIndex;
                    dataGridView1.Columns[firstlevelind1].HeaderCell.Style.BackColor =
Color.Tomato;
                    dataGridView1.Columns[firstlevelind1].HeaderText += "(1)";
                    istwofirst1 = true;
                    goodsBindingSource.ResetBindings(false);
                }
                else if (!istwosecond1 & e.ColumnIndex != firstlevelind1)
                {
                    //Если до этого был выбран только первый столбец для двухуровневой
сортировки
                    secondlevelind1 = e.ColumnIndex;
                    dataGridView1.Columns[secondlevelind1].HeaderCell.Style.BackColor =
Color.Tomato;
                    dataGridView1.Columns[secondlevelind1].HeaderText += "(2)";
                    istwosecond1 = true;
                    Singleton.TwoLevelSort(firstlevelind1, secondlevelind1, level1sort,
level2sort, Singleton.showlist);
                    goodsBindingSource.DataSource = Singleton.showlist;
                }
                else
                {
                    //Если до этого был выбрана двухуровневая сортировка
                    ChangeSelection1(firstlevelind1, 3);
                    if (istwosecond1)
                    {
                        ChangeSelection1(secondlevelind1, 3);
                        istwosecond1 = false;
                    }

                    firstlevelind1 = e.ColumnIndex;
                    dataGridView1.Columns[firstlevelind1].HeaderCell.Style.BackColor =
Color.Tomato;
                    dataGridView1.Columns[firstlevelind1].HeaderText += "(1)";
                    istwofirst1 = true;
                    goodsBindingSource.ResetBindings(false);
                }

            }
            else
            {
                // Одноуровневая сортировка по нажатию на имя столбца
                if (isonesort1)
                {
                    //Если до этого была выбрана одноуровневая сортировка
                    ChangeSelection1(lastsortind1, 1);
                    isonesort1 = false;
                }
```

```
                else
                {
                    //Если до этого была выбрана двухуровневая сортировка
                    ChangeSelection1(firstlevelind1, 3);
                    istwofirst1 = false;
                    if (istwosecond1)
                    {
                        ChangeSelection1(secondlevelind1, 3);
                        istwosecond1 = false;
                    }

                }
                sort1 *= -1;
                if (sort1 > 0)
                    dataGridView1.Columns[e.ColumnIndex].HeaderText += "↑";
                else
                    dataGridView1.Columns[e.ColumnIndex].HeaderText += "↓";
                //Собственно сортировка
                Singleton.OneLevelSort(dataGridView1.Columns[e.ColumnIndex].Index, sort1,
Singleton.showlist);

                lastsortind1 = e.ColumnIndex;
                dataGridView1.Columns[lastsortind1].HeaderCell.Style.BackColor =
Color.Aqua;
                goodsBindingSource.ResetBindings(false);
                isonesort1 = true;
            }
        }

        private void DeleteGoodBtn_Click(object sender, EventArgs e)
        {
            //Действия при нажатии на "Удалить", удаляем выбранный в таблице объект
            string row;
            if (dataGridView1.CurrentCell.Selected == true)
                row = dataGridView1.CurrentRow.Cells[0].Value.ToString();
            else
                row = dataGridView2.CurrentRow.Cells[0].Value.ToString();
            if (Singleton.newgoodlist.Find(s => s.Id == row) != null)
            {
                //Если удаляем новый товар
                Singleton.DeleteNew(row, wrh);
            }
            else
            {
                //Если удаляем существующий товар
                if (Singleton.chgoodlist.Find(s => s.Id == row) != null)
                    //Если удаляем измененный товар
                    Singleton.DeleteGenChanged(row, wrh);
                else
                    //Если удаляем НЕизмененный товар
                    Singleton.DeleteGenNotChanged(row, wrh);
            }
            RefreshShowList();
            goodsBindingSource.ResetBindings(false);
            if (showsearchlist.Count > 0)
                SearchBtn.PerformClick();
        }

        private void ChangeGoodBtn_Click(object sender, EventArgs e)
        {
            //Действия при нажатии на "Изменить", перходим на следущую форму
            string row = dataGridView1.CurrentRow.Cells[0].Value.ToString();
            Form newfrm = new GdChan(this, wrh, row);
            newfrm.ShowDialog();
            RefreshShowList();
```

```
            goodsBindingSource.ResetBindings(false);
            if (showsearchlist.Count > 0)
                SearchBtn.PerformClick();
        }

        private void ExitBtn_Click(object sender, EventArgs e)
        {
            //Сообщаем о переходе на склады
            transit = 3;
            //Закрываем форму
            this.Close();
        }

        private void Form5_FormClosing(object sender, FormClosingEventArgs e)
        {
            //Передаем координаты перехода
            oldfrm.transit = transit;
            //Переходим на форму назад
            oldfrm.Show();
        }

        private void SearchBtn_Click(object sender, EventArgs e)
        {
            //Производим поиск по указанным фильтрам
            showsearchlist = new List<Goods>();
            if (IdCmbBx.Text == "")
                IdCmbBx.Text = "*";

            if (IdCmbBx.Text != "*")
            {
                if (ModCmbBx.Text != "*")
                    showsearchlist.Add(Singleton.showlist.Find(s => s.Id.ToString() ==
IdCmbBx.Text & s.Mod.ToString() == ModCmbBx.Text));
                else
                    showsearchlist.Add(Singleton.showlist.Find(s => s.Id.ToString() ==
IdCmbBx.Text));
            }
            else if (ModCmbBx.Text != "*")
                showsearchlist.AddRange(Singleton.showlist.FindAll(s => s.Mod.ToString()
== ModCmbBx.Text));
            else
                showsearchlist.AddRange(Singleton.showlist.FindAll(s => true));

            goodsBindingSource1.DataSource = showsearchlist;
        }

        private void DataGridView2_RowEnter(object sender, DataGridViewCellEventArgs e)
        {
            //Снимаем выделение с первой таблицы
            if (showsearchlist.Count > 0)
            {
                dataGridView1.ClearSelection();
            }
        }

        private void DataGridView1_RowEnter(object sender, DataGridViewCellEventArgs e)
        {
            //Снимаем выделение со второй таблицы
            dataGridView2.ClearSelection();
        }

        private void DataGridView2_DataBindingComplete(object sender,
DataGridViewBindingCompleteEventArgs e)
        {
            // Выполняем "покрас" таблицы
```

```
            for (int i = 0; i < dataGridView2.Rows.Count; i++)
            {
                if (i % 2 == 0)
                    dataGridView2.Rows[i].DefaultCellStyle.BackColor =
Color.FromArgb(190, 224, 238);
                else
                    dataGridView2.Rows[i].DefaultCellStyle.BackColor =
Color.FromArgb(250, 246, 228);

            }
        }

        private void DataGridView2_ColumnHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
        {
            if (showsearchlist.Count > 1)
            {
                    // Одноуровневая сортировка по нажатию на имя столбца
                        //Если до этого была выбрана одноуровневая сортировка
                    ChangeSelection2(lastsortind2, 1);
                        //Если до этого была выбрана двухуровневая сортировка

                    sort2 *= -1;
                    if (sort2 > 0)
                        dataGridView2.Columns[e.ColumnIndex].HeaderText += "↑";
                    else
                        dataGridView2.Columns[e.ColumnIndex].HeaderText += "↓";
                    //Собственно сортировка
                    Singleton.OneLevelSort(dataGridView2.Columns[e.ColumnIndex].Index,
sort2, showsearchlist);

                    lastsortind2 = e.ColumnIndex;
                    dataGridView2.Columns[lastsortind2].HeaderCell.Style.BackColor =
Color.Aqua;
                    goodsBindingSource1.ResetBindings(false);

            }
        }

        private void ОбАвтореToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Программу разработал студент 2-го курса группы ПИ18-3 Комлев
Н.П.",
                "Версия 1.0", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void ОПрограммеToolStripMenuItem_Click(object sender, EventArgs e)
        {
            System.Diagnostics.Process.Start("notepad.exe", "info.txt");
        }

        private void ChangeSelection1(int i, int z)
        {
            var x = dataGridView1.Columns[i].HeaderText;
            dataGridView1.Columns[i].HeaderText = x.Substring(0, x.Length - z);
            dataGridView1.Columns[i].HeaderCell.Style.BackColor = DefaultBackColor;
        }

        private void ChangeSelection2(int i, int z)
        {
            var x = dataGridView2.Columns[i].HeaderText;
            dataGridView2.Columns[i].HeaderText = x.Substring(0, x.Length - z);
            dataGridView2.Columns[i].HeaderCell.Style.BackColor = DefaultBackColor;
        }
```

```
        private void SortBtn_Click(object sender, EventArgs e)
        {
            Form newfrm = new SortF(null, this);
            newfrm.ShowDialog();
        }

        private void УправлениеСкладToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //Сообщаем о переходе на склады
            transit = 3;
            //Закрываем форму
            this.Close();
        }

        private void УчетToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //Сообщаем о переходе в учет
            transit = 1;
            //Закрываем форму
            this.Close();
        }

        private void ГлавнаяToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //Сообщаем о переходе на главную
            transit = 0;
            //Закрываем форму
            this.Close();
        }
    }
}
```

Файл ChAdd.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace KR
{
    public partial class ChAdd : Form
    {
        //Форма добавления объектов
        //Можно перейти на формы добавления товаров, и других объектов (категория,
модель, производитель, поставщик) и вернуться на форму Управления данными: Товары
        MngGds oldfrm;
        string wrh;

        public ChAdd(MngGds oldfrm_, string wrh_)
        {
            InitializeComponent();
            //Получаем ссылку на предыдущую форму
            oldfrm = oldfrm_;
            //Получаем информацию о том, с каким складом работаем
            wrh = wrh_;
        }

        private void Form8_Load(object sender, EventArgs e)
        {
```

```
        }

        private void AddGdBtn_Click(object sender, EventArgs e)
        {
            //Действия по нажатию на "Добавить", переход на новую форму
            Form newfrm = new GdAdd(oldfrm, wrh);
            newfrm.ShowDialog();
            DialogResult = DialogResult.OK;
        }

        private void ExitBtn_Click(object sender, EventArgs e)
        {
            DialogResult = DialogResult.OK;
        }

        private void ModBtn_Click(object sender, EventArgs e)
        {
            Form newfrm = new ModAdd();
            newfrm.ShowDialog();
            DialogResult = DialogResult.OK;
        }

        private void CtgBtn_Click(object sender, EventArgs e)
        {
            Form newfrm = new ObjAdd(oldfrm, "Категория");
            newfrm.ShowDialog();
            DialogResult = DialogResult.OK;
        }

        private void ProBtn_Click(object sender, EventArgs e)
        {
            Form newfrm = new ObjAdd(oldfrm, "Производитель");
            newfrm.ShowDialog();
            DialogResult = DialogResult.OK;
        }

        private void DisBtn_Click(object sender, EventArgs e)
        {
            Form newfrm = new ObjAdd(oldfrm, "Поставщик");
            newfrm.ShowDialog();
            DialogResult = DialogResult.OK;
        }
    }
}
```

Файл GdAdd.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace KR
{
    public partial class GdAdd : Form
    {
        //Форма добавления товара
        //Можно возвратиться на форму Управления данными: Товары
        MngGds oldfrm;
```

```
        string wrh;

        public GdAdd(MngGds oldfrm_, string wrh_)
        {
            InitializeComponent();
            //Получаем ссылку на предыдущую форму
            oldfrm = oldfrm_;
            //Получаем информацию о том, с каким складом работаем
            wrh = wrh_;
        }

        private void Form6_Load(object sender, EventArgs e)
        {
            label1.Text += wrh;
            //Заполняем комбобоксы для категорий и поставщиков
            for (int i = 0; i < Singleton.ctglist.Count; i++)
            {
                CtgCmbBx.Items.Add(Singleton.ctglist[i].ToString());
            }
            for (int i = 0; i < Singleton.dislist.Count; i++)
            {
                DisCmbBx.Items.Add(Singleton.dislist[i].ToString());
            }
        }

        private void CtgCmbBx_TextChanged(object sender, EventArgs e)
        {
            //При выборе значения категории, вызываем функцию появления следующего
комбобокса
            if (!CtgCmbBx.Items.Contains(CtgCmbBx.Text)) //Защита от пользовательского
ввода
                CtgCmbBx.Text = CtgCmbBx.Items[0].ToString();
            if (CtgCmbBx.Text.Length > 0)
                Pro_Show(CtgCmbBx.Text);
        }

        public void Pro_Show(string ctg_)
        {
            //Функция появления комбобокса производителей
            //Принимает значение номера категории
            label4.Visible = true;
            ProCmbBx.Visible = true;
            ProCmbBx.Items.Clear();
            ProCmbBx.Text = "";
            //Заполняем комбобокс производителями
            for (int i = 0; i < Singleton.prolist.Count; i++)
            {
                ProCmbBx.Items.Add(Singleton.prolist[i].ToString());
            }
        }

        private void ProCmbBx_TextChanged(object sender, EventArgs e)
        {
            //При выборе значения производителя вызываем функцию появления слующего
комбобокса
            //В случае, если значение пустое, прячем следующий комбобокс
            if (!ProCmbBx.Items.Contains(ProCmbBx.Text) & ProCmbBx.Items.Count > 0)
                ProCmbBx.Text = ProCmbBx.Items[0].ToString();
            if (ProCmbBx.Text.Length > 0)
                Mod_Show(ProCmbBx.Text, CtgCmbBx.Text);
            else Mod_Hide();
        }

        public void Mod_Show(string pro_, string ctg_)
        {
```

```
            //Функция появления комобобокса моделей
            //Принимает значения номер производителя и номера категории
            OtherHide();
            label3.Visible = true;
            ModCmbBx.Visible = true;
            ModCmbBx.Items.Clear();
            ModCmbBx.Text = "";
            //Заполняем комбобокс в соответствие с выбранными категориями и
производителем
            for (int i = 0; i < Singleton.modlist.Count; i++)
            {
                if ((Singleton.modlist[i].Prod.ToString() == pro_) &
(Singleton.modlist[i].Categ.ToString() == ctg_))
                    ModCmbBx.Items.Add(Singleton.modlist[i].ToString());
            }
        }

        public void Mod_Hide()
        {
            //Функция скрытия комобобокса моделей
            label3.Visible = false;
            ModCmbBx.Visible = false;
            ModCmbBx.Items.Clear();
            ModCmbBx.Text = "";
            OtherHide();
        }

        private void ModCmbBx_TextChanged(object sender, EventArgs e)
        {
            //При выборе значения модели, показываем все остальные элементы
            if (!ModCmbBx.Items.Contains(ModCmbBx.Text) & ModCmbBx.Items.Count > 0)
                ModCmbBx.Text = ModCmbBx.Items[0].ToString();
            if (ModCmbBx.Text.Length > 0)
            {
                OtherShow();
            }
        }

        private void DisCmbBx_TextChanged(object sender, EventArgs e)
        {
            //Проверка заполненности комбобокса поставщиков
            if (!DisCmbBx.Items.Contains(DisCmbBx.Text))
                DisCmbBx.Text = DisCmbBx.Items[0].ToString();
            ConfirmCheck();
        }

        private void WthTxt_TextChanged(object sender, EventArgs e)
        {
            //Проверка заполненности поля цены
            ConfirmCheck();
        }

        public void ConfirmCheck()
        {
            //Если все заполнено, разрешить кнопку подтверждения
            if ((WthTxt.Text.Length > 0) & (DisCmbBx.Text.Length > 0))
                YesBtn.Enabled = true;
            else
                YesBtn.Enabled = false;
        }

        public void OtherHide()
        {
            //Функция скрывания остальных элементов
            WthTxt.Visible = false;
```

```
            label5.Visible = false;
            WthTxt.Text = "";
            DatePick.Visible = false;
            label6.Visible = false;
            StcCheck.Visible = false;
            label7.Visible = false;
            DisCmbBx.Visible = false;
            DisCmbBx.Text = "";
            label8.Visible = false;
        }

        public void OtherShow()
        {
            //Функция появления всех элементов
            WthTxt.Visible = true;
            label5.Visible = true;
            DatePick.Visible = true;
            label6.Visible = true;
            StcCheck.Visible = true;
            label7.Visible = true;
            DisCmbBx.Visible = true;
            label8.Visible = true;
        }

        private void YesBtn_Click(object sender, EventArgs e)
        {
            //Проверка на уверенность о внесении изменений
            DialogResult res = MessageBox.Show(
                "Уверены, что хотите внести изменения?",
                "Сообщение",
                MessageBoxButtons.YesNo,
                MessageBoxIcon.Information);
            if (res == DialogResult.Yes)
            {
                try
                {
                    //Создаем новый айдентификатор на основе уже имеющихся данных
                    string keyy = wrh + Singleton.modlist.Find(s => s.ToString() ==
ModCmbBx.Text).Id.Substring(3);
                    string idd = "";
                    if (Singleton.good_id.Keys.Contains(keyy))
                    {
                        Singleton.good_id[keyy] += 1;
                        idd = keyy + Singleton.good_id[keyy].ToString().PadLeft(5, '0');
                    }
                    else
                    {
                        Singleton.good_id.Add(keyy, 1);
                        idd = keyy + Singleton.good_id[keyy].ToString().PadLeft(5, '0');
                    }
                    //Создаем новый экземпляр класса
                    var new_gd = new Goods(idd,
                                           Singleton.ctglist.Find(s => s.ToString() ==
CtgCmbBx.Text),
                                           Singleton.modlist.Find(s => s.ToString() ==
ModCmbBx.Text),
                                           Singleton.prolist.Find(s => s.ToString() ==
ProCmbBx.Text),
                                           Convert.ToDouble(WthTxt.Text),
                                           DatePick.Value,
                                           StcCheck.Checked,
                                           Singleton.dislist.Find(s => s.ToString() ==
DisCmbBx.Text),
                                           Singleton.wrhlist.Find(s => s.ToString() ==
wrh));
```

```
                    //Помещаем в общий список и список новых товаров
                    Singleton.wrhlist.Find(s => s.ToString() ==
wrh).GoodsLst.Add(new_gd);
                    Singleton.newgoodlist.Add(new_gd);
                    DialogResult = DialogResult.OK;
                }
                catch
                {
                    //Если ошибка из-за некорректно введенных данных
                    MessageBox.Show(
                        "Некорректный ввод",
                        "Сообщение",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Information,
                        MessageBoxDefaultButton.Button1);
                }
            }
        }

        private void CancelBtn_Click(object sender, EventArgs e)
        {
            //Закрываем форму
            DialogResult = DialogResult.OK;
        }
    }
}
```

Файл GdChan.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace KR
{
    public partial class GdChan : Form
    {
        //Форма изменения товара
        //Можно возвратиться на форму Управления данными: Товары
        MngGds oldfrm;
        string wrh;
        string row;

        public GdChan(MngGds oldfrm_, string wrh_, string row_)
        {
            InitializeComponent();
            //Получаем ссылку на предыдущую форму
            oldfrm = oldfrm_;
            //Получаем информацию о том, с каким складом и каким объектом работаем
            wrh = wrh_;
            row = row_;
        }

        private void Form7_Load(object sender, EventArgs e)
        {
            //Заполняем все элементы исходными данными выбранного объекта
            label1.Text += wrh;
            var x = Singleton.wrhlist.Find(s => s.ToString() == wrh).GoodsLst.Find(u =>
u.Id == row);
```

```
            for (int i = 0; i < Singleton.ctglist.Count; i++)
            {
                CtgCmbBx.Items.Add(Singleton.ctglist[i].ToString());
            }

            CtgCmbBx.Text = x.Categ.ToString();
            ProCmbBx.Text = x.Prod.ToString();
            ModCmbBx.Text = x.Mod.ToString();
            WthTxt.Text = x.Worth.ToString();
            DatePick.Value = x.ArDate;
            StcCheck.Checked = x.InStock;
            DisCmbBx.Text = x.Distr.ToString();

            for (int i = 0; i < Singleton.dislist.Count; i++)
            {
                DisCmbBx.Items.Add(Singleton.dislist[i].ToString());
            }
        }

        private void CtgCmbBx_TextChanged(object sender, EventArgs e)
        {
            //При выборе значения категории, вызываем функцию появления следующего
комбобокса
            if (!CtgCmbBx.Items.Contains(CtgCmbBx.Text)) //Защита от пользовательского
ввода
                CtgCmbBx.Text = CtgCmbBx.Items[0].ToString();
            if (CtgCmbBx.Text.Length > 0)
                Pro_Show(CtgCmbBx.Text);
        }

        public void Pro_Show(string ctg_)
        {
            //Функция появления комбобокса производителей
            //Принимает значение номера катогории
            label4.Visible = true;
            ProCmbBx.Visible = true;
            ProCmbBx.Items.Clear();
            ProCmbBx.Text = "";
            //Заполняем комбобокс производителями
            for (int i = 0; i < Singleton.prolist.Count; i++)
            {
                ProCmbBx.Items.Add(Singleton.prolist[i].ToString());
            }
        }

        private void ProCmbBx_TextChanged(object sender, EventArgs e)
        {
            //При выборе значения производителя вызываем функцию появления слующего
комбобокса
            //В случае, если значение пустое, прячем следующий комбобокс
            if (!ProCmbBx.Items.Contains(ProCmbBx.Text) & ProCmbBx.Items.Count > 0)
                ProCmbBx.Text = ProCmbBx.Items[0].ToString();
            if (ProCmbBx.Text.Length > 0)
                Mod_Show(ProCmbBx.Text, CtgCmbBx.Text);
            else Mod_Hide();
        }

        public void Mod_Show(string pro_, string ctg_)
        {
            //Функция появления комобобокса моделей
            //Принимает значения номер производителя и номера категории
            OtherHide();
            label3.Visible = true;
            ModCmbBx.Visible = true;
            ModCmbBx.Items.Clear();
```

```
            ModCmbBx.Text = "";
            //Заполняем комбобокс в соответствие с выбранными категориями и
производителем
            for (int i = 0; i < Singleton.modlist.Count; i++)
            {
                if ((Singleton.modlist[i].Prod.ToString() == pro_) &
(Singleton.modlist[i].Categ.ToString() == ctg_))
                    ModCmbBx.Items.Add(Singleton.modlist[i].ToString());
            }
        }

        public void Mod_Hide()
        {
            //Функция скрытия комобобокса моделей
            label3.Visible = false;
            ModCmbBx.Visible = false;
            ModCmbBx.Items.Clear();
            ModCmbBx.Text = "";
            OtherHide();
        }

        private void ModCmbBx_TextChanged(object sender, EventArgs e)
        {
            //При выборе значения модели, показываем все остальные элементы
            if (!ModCmbBx.Items.Contains(ModCmbBx.Text) & ModCmbBx.Items.Count > 0)
                ModCmbBx.Text = ModCmbBx.Items[0].ToString();
            if (ModCmbBx.Text.Length > 0)
            {
                OtherShow();
            }
        }

        private void DisCmbBx_TextChanged(object sender, EventArgs e)
        {
            //Проверка заполненности комбобокса поставщиков
            if (!DisCmbBx.Items.Contains(DisCmbBx.Text) & DisCmbBx.Items.Count > 0)
                DisCmbBx.Text = DisCmbBx.Items[0].ToString();
            ConfirmCheck();
        }

        private void WthTxt_TextChanged(object sender, EventArgs e)
        {
            //Проверка заполненности поля цены
            ConfirmCheck();
        }

        public void ConfirmCheck()
        {
            //Если все заполнено, разрешить кнопку подтверждения
            if ((WthTxt.Text.Length > 0) & (DisCmbBx.Text.Length > 0))
                YesBtn.Enabled = true;
            else
                YesBtn.Enabled = false;
        }

        public void OtherHide()
        {
            //Функция скрывания остальных элементов
            WthTxt.Visible = false;
            label5.Visible = false;
            WthTxt.Text = "";
            DatePick.Visible = false;
            label6.Visible = false;
            StcCheck.Visible = false;
            label7.Visible = false;
```

```
            DisCmbBx.Visible = false;
            DisCmbBx.Text = "";
            label8.Visible = false;
        }

        public void OtherShow()
        {
            //Функция появления всех элементов
            WthTxt.Visible = true;
            label5.Visible = true;
            DatePick.Visible = true;
            label6.Visible = true;
            StcCheck.Visible = true;
            label7.Visible = true;
            DisCmbBx.Visible = true;
            label8.Visible = true;
        }

        private void YesBtn_Click(object sender, EventArgs e)
        {
            //Проверка на уверенность о внесении изменений
            DialogResult res = MessageBox.Show(
                "Уверены, что хотите внести изменения?",
                "Сообщение",
                MessageBoxButtons.YesNo,
                MessageBoxIcon.Information);
            if (res == DialogResult.Yes)
            {
                try
                {
                    if (Singleton.newgoodlist.Contains(Singleton.wrhlist.Find(s =>
s.ToString() == wrh).GoodsLst.Find(u => u.Id == row)))
                    {
                        //Если меняем новый товар
                        Singleton.ChangeNew(row,
                                            Singleton.ctglist.Find(s => s.ToString() ==
CtgCmbBx.Text),
                                            Singleton.prolist.Find(s => s.ToString() ==
ProCmbBx.Text),
                                            Singleton.modlist.Find(s => s.ToString() ==
ModCmbBx.Text),
                                            Convert.ToDouble(WthTxt.Text),
                                            DatePick.Value,
                                            StcCheck.Checked,
                                            Singleton.dislist.Find(s => s.ToString() ==
DisCmbBx.Text));
                        DialogResult = DialogResult.OK;
                    }
                    else
                    {
                        //Если меняем существующий товар
                        Singleton.ChangeGen(row,
                                            Singleton.ctglist.Find(s => s.ToString() ==
CtgCmbBx.Text),
                                            Singleton.prolist.Find(s => s.ToString() ==
ProCmbBx.Text),
                                            Singleton.modlist.Find(s => s.ToString() ==
ModCmbBx.Text),
                                            Convert.ToDouble(WthTxt.Text),
                                            DatePick.Value,
                                            StcCheck.Checked,
                                            Singleton.dislist.Find(s => s.ToString() ==
DisCmbBx.Text),
                                            Singleton.wrhlist.Find(s => s.ToString() ==
wrh));
```

```
                    DialogResult = DialogResult.OK;
                }

            }
            catch
            {
                //Если ошибка из-за некорректно введенных данных
                MessageBox.Show(
                    "Некорректный ввод",
                    "Сообщение",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information,
                    MessageBoxDefaultButton.Button1);
            }
        }
    }

    private void CancelBtn_Click(object sender, EventArgs e)
    {
        //Закрываем форму
        DialogResult = DialogResult.OK;
    }
  }
}
```

Файл ObjAdd.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace KR
{
    public partial class ObjAdd : Form
    {
        //Форма добавления объекта(катег, произв, поставщ)
        //Можно возвратиться на форму Управления данными: Товары
        MngGds oldfrm;
        string addobj;

        public ObjAdd(MngGds oldfrm_, string addobj_)
        {
            InitializeComponent();
            //Получаем ссылку на предыдущую форму
            oldfrm = oldfrm_;
            //Получаем информацию о том, какой объект добавляем
            addobj = addobj_;
        }

        private void Form9_Load(object sender, EventArgs e)
        {
            //Заполняем лэйбл названием объекта
            label1.Text += addobj;
        }

        private void TextBox1_TextChanged(object sender, EventArgs e)
        {
            //Проверка на пустоту текстбокса
            if (textBox1.Text.Length <= 0)
```

```
                YesBtn.Enabled = false;
            else YesBtn.Enabled = true;
        }

        private void CancelBtn_Click(object sender, EventArgs e)
        {
            DialogResult = DialogResult.OK;
        }

        private void YesBtn_Click(object sender, EventArgs e)
        {
            //Добавление нового объекта
            DialogResult res = MessageBox.Show(
                "Уверены, что хотите внести изменения?",
                "Сообщение",
                MessageBoxButtons.YesNo,
                MessageBoxIcon.Information);
            if (res == DialogResult.Yes)
            {
                string keyy;
                if (addobj == "Категория")
                {
                    keyy = "ctg" + (Singleton.ctg_id + 1).ToString();
                    Singleton.ctg_id += 1;
                    Singleton.newctglist.Add(new Category(keyy, textBox1.Text));
                    Singleton.ctglist.Add(new Category(keyy, textBox1.Text));
                }
                else if (addobj == "Производитель")
                {
                    keyy = "pro" + (Singleton.pro_id + 1).ToString().PadLeft(3, '0');
                    Singleton.pro_id += 1;
                    Singleton.newprolist.Add(new Producer(keyy, textBox1.Text));
                    Singleton.prolist.Add(new Producer(keyy, textBox1.Text));
                }
                else if (addobj == "Поставщик")
                {
                    keyy = "dis" + (Singleton.dis_id + 1).ToString().PadLeft(2, '0');
                    Singleton.dis_id += 1;
                    Singleton.newdislist.Add(new Distributor(keyy, textBox1.Text));
                    Singleton.dislist.Add(new Distributor(keyy, textBox1.Text));
                }
                DialogResult = DialogResult.OK;
            }
        }
    }
}
```

Файл ModAdd.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace KR
{
    public partial class ModAdd : Form
    {
        //Форма добавления модели
        //Можно возвратиться на форму Управления данными: Товары
```

```
        public ModAdd()
        {
            InitializeComponent();
        }

        public void CheckFill()
        {
            //Функция проверки заполненности всех полей
            if (textBox1.Text.Length > 0 & CtgCmbBx.Text.Length > 0 &
ProCmbBx.Text.Length > 0)
                YesBtn.Enabled = true;
            else
                YesBtn.Enabled = false;
        }

        private void CancelBtn_Click(object sender, EventArgs e)
        {
            //Закрываем форму
            DialogResult = DialogResult.OK;
        }

        private void TextBox1_TextChanged(object sender, EventArgs e)
        {
            CheckFill();
        }

        private void CtgCmbBx_TextChanged(object sender, EventArgs e)
        {
            if (!CtgCmbBx.Items.Contains(CtgCmbBx.Text)) //Защита от пользовательского
ввода
                CtgCmbBx.Text = CtgCmbBx.Items[0].ToString();
            CheckFill();
        }

        private void ProCmbBx_TextChanged(object sender, EventArgs e)
        {
            if (!ProCmbBx.Items.Contains(ProCmbBx.Text) & ProCmbBx.Items.Count >
0)//Защита от пользовательского ввода
                ProCmbBx.Text = ProCmbBx.Items[0].ToString();
            CheckFill();
        }

        private void Form10_Load(object sender, EventArgs e)
        {
            //Заполняем комбобоксы данными
            for (int i = 0; i < Singleton.ctglist.Count; i++)
            {
                CtgCmbBx.Items.Add(Singleton.ctglist[i].ToString());
            }
            for (int i = 0; i < Singleton.prolist.Count; i++)
            {
                ProCmbBx.Items.Add(Singleton.prolist[i].ToString());
            }
        }

        private void YesBtn_Click(object sender, EventArgs e)
        {
            //Функция добавления изменения
            DialogResult res = MessageBox.Show(
                "Уверены, что хотите внести изменения?",
                "Сообщение",
                MessageBoxButtons.YesNo,
                MessageBoxIcon.Information);
            if (res == DialogResult.Yes)
            {
```

```
                string keyy;
                keyy = "mod" + (Singleton.mod_id + 1).ToString().PadLeft(3, '0');
                Singleton.mod_id += 1;
                Singleton.newmodlist.Add(new Model(keyy, textBox1.Text,
Singleton.ctglist.Find(s => s.ToString() == CtgCmbBx.Text), Singleton.prolist.Find(s =>
s.ToString() == ProCmbBx.Text)));
                Singleton.modlist.Add(new Model(keyy, textBox1.Text,
Singleton.ctglist.Find(s => s.ToString() == CtgCmbBx.Text), Singleton.prolist.Find(s =>
s.ToString() == ProCmbBx.Text)));
                DialogResult = DialogResult.OK;
            }
        }
    }
}
```

Файл SortF.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace KR
{
    public partial class SortF : Form
    {
        //Если форма вызывается из второй формы
        AccountF oldfrm2;
        //Если форма вызывается из первой формы
        MngGds oldfrm5;
        public SortF(AccountF oldfrm2_, MngGds oldfrm5_)
        {
            InitializeComponent();
            oldfrm2 = oldfrm2_;
            oldfrm5 = oldfrm5_;
        }

        private void Form11_Load(object sender, EventArgs e)
        {
            if (oldfrm5 is null)
            {
                ParLevel1.Items.Add("По убыванию");
                ParLevel1.Items.Add("По возрастанию");
                ParLevel1.Text = oldfrm2.level1sort;
                ParLevel2.Items.Add("По убыванию");
                ParLevel2.Items.Add("По возрастанию");
                ParLevel2.Text = oldfrm2.level2sort;
            }
            else
            {
                ParLevel1.Items.Add("По убыванию");
                ParLevel1.Items.Add("По возрастанию");
                ParLevel1.Text = oldfrm5.level1sort;
                ParLevel2.Items.Add("По убыванию");
                ParLevel2.Items.Add("По возрастанию");
                ParLevel2.Text = oldfrm5.level2sort;
            }

        }
```

```
        private void ParLevel1_TextChanged(object sender, EventArgs e)
        {
            //Проверка для ограничения пользовательского ввода
            if (!ParLevel1.Items.Contains(ParLevel1.Text))
                ParLevel1.Text = ParLevel1.Items[0].ToString();
        }

        private void ParLevel2_TextChanged(object sender, EventArgs e)
        {
            //Проверка для ограничения пользовательского ввода
            if (!ParLevel2.Items.Contains(ParLevel2.Text))
                ParLevel2.Text = ParLevel2.Items[0].ToString();
        }

        private void YesBtn_Click(object sender, EventArgs e)
        {
            DialogResult = DialogResult.OK;
        }

        private void Form11_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (oldfrm5 is null)
            {
                oldfrm2.level1sort = ParLevel1.Text;
                oldfrm2.level2sort = ParLevel2.Text;
            }
            else
            {
                oldfrm5.level1sort = ParLevel1.Text;
                oldfrm5.level2sort = ParLevel2.Text;
            }
        }
    }
}
```

Файл Class1.cs:

```
using System;
using System.Data.SqlClient;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Collections;
using System.Net;
using System.IO;

namespace KR
{
    public class Warehouse
    {
        //Класс "склад". Содержит три свойства:
        // Id - строковое значение, идентификационный номер склада.
        // Address - строковое значение адреса склада.
        // GoodsLst - список содержащихся на складе товаров.

        public string Id { get; set; }
        public string Address { get; set; }
        public List<Goods> GoodsLst { get; set; }


        public Warehouse(string id, object addr, List<Goods> goods)
```

```
    {
        Id = id;
        if (addr is string)
            Address = (string)addr;
        GoodsLst = goods;
    }

    public override string ToString()
    {
        return string.Format("{0}", Id.Substring(3));
    }

}

public class Distributor
{
    // Класс "поставщик". Содержит два свойства:
    // Id - целочисленное значение, идентификационный номер поставщика.
    // Name - строковое название компании поставщика.

    public string Id { get; set; }
    public string Name { get; set; }

    public Distributor(string id, string name)
    {
        Id = id;
        Name = name;
    }

    public override string ToString()
    {
        return string.Format("{0}", Name);
    }
}

public class Category
{
    //Класс "категорий". Содержит два свойства:
    //Id - строковое значение, идентификационный номер категории товара.
    //Name - строковое название категории.

    public string Id { get; set; }
    public string Name { get; set; }

    public Category(string id, string name)
    {
        Id = id;
        Name = name;
    }

    public override string ToString()
    {
        return string.Format("{0}", Name);
    }
}

public class Producer
{
    //Класс "производителей". Содержит два свойства:
    // Id - строковое значение, идентификационный номер производителя.
    // Name - строковое название производителя.

    public string Id { get; set; }
    public string Name { get; set; }
```

```
        public Producer(string id, string name)
        {
            Id = id;
            Name = name;
        }

        public override string ToString()
        {
            return string.Format("{0}", Name);
        }
    }

    public class Model
    {
        // Класс "моделей". Содержит четыре свойства:
        // Id - строковое значение, идентификационный номер модели.
        // Name - строковое название модели.
        // Ctg - экземпляр класса категории модели.
        // Prod - экземпляр класса производителя.

        public string Id { get; set; }
        public string Name { get; set; }
        public Category Categ { get; set; }
        public Producer Prod { get; set; }

        public Model(string id, string name, Category ctg, Producer prod)
        {
            Id = id;
            Name = name;
            Categ = ctg;
            Prod = prod;
        }

        public override string ToString()
        {
            return string.Format("{0}", Name);
        }
    }

    public class Goods
    {
        //Класс "товар". Содержит восемь свойств:
        // Id - строковое значение, идентификационный номер товара.
        // Categ - экземпляр класса категории.
        // Model - экземпляр класса модели.
        // Producer - экземпляр класса производителя.
        // Worth - вещественное значение цены.
        // ArDate - дата поступления на склад.
        // InStock - булевское обозначение: "в наличии".
        // Distr - экземпляр класса поставщика.

        public string Id { get; set; }
        public Category Categ { get; set; }
        public Model Mod { get; set; }
        public Producer Prod { get; set; }
        public double Worth { get; set; }
        public DateTime ArDate { get; set; }
        public bool InStock { get; set; }
        public Distributor Distr { get; set; }
        public Warehouse WrhId { get; set; }

        public Goods(string id, Category categ, Model mod, Producer prod, double worth,
DateTime ardate, bool instock, Distributor distr, Warehouse wrhid)
        {
            Id = id;
```

```
            Categ = categ;
            Mod = mod;
            Prod = prod;
            Worth = worth;
            ArDate = ardate;
            InStock = instock;
            Distr = distr;
            WrhId = wrhid;
        }


        public override string ToString()
        {
            return string.Format("Идентификационный номер: {0}, \nкатегория товара: {1},
\nмодель: {2}," +
                " \nпроизводитель: {3}, \nцена: {4} р.\n", Id, Categ, Mod, Prod, Worth);
        }
    }

    public static class Singleton
    {
        //Статический класс предназначен для хранения всех данных,
        //над которыми проводятся операции (хранение, изменение, удаление, поиск)

        //Списки, содержащие все экземпляры всех классов
        public static List<Warehouse> wrhlist = new List<Warehouse>();
        public static List<Distributor> dislist = new List<Distributor>();
        public static List<Category> ctglist = new List<Category>();
        public static List<Producer> prolist = new List<Producer>();
        public static List<Model> modlist = new List<Model>();
        public static List<Goods> showlist = new List<Goods>();

        //Переменные, хранящие последнее значение идентификатора для каждого класса
        public static int wr_id = 0;
        public static int dis_id = 0;
        public static int ctg_id = 0;
        public static int pro_id = 0;
        public static int mod_id = 0;
        public static Dictionary<string, int> good_id = new Dictionary<string, int>();

        //Списки, содержащие элементы, которые ожидают, чтобы их добавили в БД, изменили
и удалили из нее соответственно
        public static List<Goods> newgoodlist = new List<Goods>();
        public static List<Goods> chgoodlist = new List<Goods>();
        public static List<Goods> delgoodlist = new List<Goods>();

        public static List<Model> newmodlist = new List<Model>();
        public static List<Category> newctglist = new List<Category>();
        public static List<Producer> newprolist = new List<Producer>();
        public static List<Distributor> newdislist = new List<Distributor>();

        //Путь для подключения к серверу
        public static string connectionString = @"Data Source=.\SQLEXPRESS;Initial
Catalog=Storage;Integrated Security=True";

        static Singleton()
        {
            // Заполняем списки данными из базы
            using (SqlConnection connection = new SqlConnection(connectionString))
            {
                connection.Open();

                // Заполняем список Складов
                SqlCommand command = new SqlCommand("SELECT * FROM Warehouses",
connection);
```

```
                SqlDataReader reader = command.ExecuteReader();

                if (reader.HasRows) // если есть данные
                {
                    while (reader.Read()) // построчно считываем данные
                    {
                        wrhlist.Add(new Warehouse((string)reader.GetValue(0),
reader.GetValue(1), new List<Goods>()));
                    }
                }
                reader.Close();

                // Заполняем список поставщиков
                SqlCommand command2 = new SqlCommand("SELECT * FROM Distributors",
connection);
                SqlDataReader reader2 = command2.ExecuteReader();

                if (reader2.HasRows) // если есть данные
                {
                    while (reader2.Read()) // построчно считываем данные
                    {
                        dislist.Add(new Distributor((string)reader2.GetValue(0),
(string)reader2.GetValue(1)));
                    }
                }
                reader2.Close();

                // Заполняем список категорий
                SqlCommand command3 = new SqlCommand("SELECT * FROM Categories",
connection);
                SqlDataReader reader3 = command3.ExecuteReader();

                if (reader3.HasRows) // если есть данные
                {
                    while (reader3.Read()) // построчно считываем данные
                    {
                        ctglist.Add(new Category((string)reader3.GetValue(0),
(string)reader3.GetValue(1)));
                    }
                }
                reader3.Close();

                // Заполняем список производителей
                SqlCommand command4 = new SqlCommand("SELECT * FROM Producers",
connection);
                SqlDataReader reader4 = command4.ExecuteReader();

                if (reader4.HasRows) // если есть данные
                {
                    while (reader4.Read()) // построчно считываем данные
                    {
                        prolist.Add(new Producer((string)reader4.GetValue(0),
(string)reader4.GetValue(1)));
                    }
                }
                reader4.Close();

                // Заполняем список моделей
                SqlCommand command5 = new SqlCommand("SELECT * FROM Models", connection);
                SqlDataReader reader5 = command5.ExecuteReader();

                if (reader5.HasRows) // если есть данные
                {
                    while (reader5.Read()) // построчно считываем данные
                    {
```

```
                        modlist.Add(new Model((string)reader5.GetValue(0),
(string)reader5.GetValue(1),
                                              ctglist.Find(s => s.Id ==
(string)reader5.GetValue(2)),
                                              prolist.Find(s => s.Id ==
(string)reader5.GetValue(3))));
                    }
                }
                reader5.Close();

                // Заполняем список товаров
                SqlCommand commandn = new SqlCommand("SELECT * FROM Goods", connection);
                SqlDataReader readern = commandn.ExecuteReader();

                if (readern.HasRows) // если есть данные
                {
                    while (readern.Read()) // построчно считываем данные
                    {
                        wrhlist.Find(s => s.Id ==
(string)readern.GetValue(6)).GoodsLst.Add(new Goods((string)readern.GetValue(0),

modlist.Find(s => s.Id == (string)readern.GetValue(1)).Categ,

modlist.Find(s => s.Id == (string)readern.GetValue(1)),

modlist.Find(s => s.Id == (string)readern.GetValue(1)).Prod,

(double)readern.GetValue(2),

(DateTime)readern.GetValue(3),

(bool)readern.GetValue(4),

dislist.Find(s => s.Id == (string)readern.GetValue(5)),

wrhlist.Find(s => s.Id == (string)readern.GetValue(6))));
                    }
                }
                readern.Close();

                RefreshShowList();

                // Заполняем переменные с последними значениями идентификационных номеров
                for (int i = 0; i < wrhlist.Count; i++)
                {
                    if (Convert.ToInt32(wrhlist[i].Id.Substring(3)) > wr_id)
                        wr_id = Convert.ToInt32(wrhlist[i].Id.Substring(3));
                }
                for (int i = 0; i < dislist.Count; i++)
                {
                    if (Convert.ToInt32(dislist[i].Id.Substring(3)) > dis_id)
                        dis_id = Convert.ToInt32(dislist[i].Id.Substring(3));
                }
                for (int i = 0; i < ctglist.Count; i++)
                {
                    if (Convert.ToInt32(ctglist[i].Id.Substring(3)) > ctg_id)
                        ctg_id = Convert.ToInt32(ctglist[i].Id.Substring(3));
                }
                for (int i = 0; i < prolist.Count; i++)
                {
                    if (Convert.ToInt32(prolist[i].Id.Substring(3)) > pro_id)
                        pro_id = Convert.ToInt32(prolist[i].Id.Substring(3));
                }
                for (int i = 0; i < modlist.Count; i++)
                {
```

```
                    if (Convert.ToInt32(modlist[i].Id.Substring(3)) > mod_id)
                        mod_id = Convert.ToInt32(modlist[i].Id.Substring(3));
                }

                for (int i = 0; i < wrhlist.Count; i++)
                {
                    for (int j = 0; j < wrhlist[i].GoodsLst.Count; j++)
                    {
                        if (good_id.ContainsKey(wrhlist[i].GoodsLst[j].Id.Substring(0,
5)))
                        {
                            if (good_id[wrhlist[i].GoodsLst[j].Id.Substring(0, 5)] <
Convert.ToInt32(wrhlist[i].GoodsLst[j].Id.Substring(5, 5)))
                                good_id[wrhlist[i].GoodsLst[j].Id.Substring(0, 5)] =
Convert.ToInt32(wrhlist[i].GoodsLst[j].Id.Substring(5, 5));
                            else { };
                        }
                        else good_id.Add(wrhlist[i].GoodsLst[j].Id.Substring(0, 5),
Convert.ToInt32(wrhlist[i].GoodsLst[j].Id.Substring(5, 5)));
                    }
                }
            }
        }

        public static void RefreshShowList()
        {
            //Функция обновления шоулиста, для вывода всех товаров со всех складов
            showlist.Clear();
            // Заполняем шоулист  (спсиок товаров для вывода)
            for (int i = 0; i < wrhlist.Count; i++)
            {
                for (int j = 0; j < wrhlist[i].GoodsLst.Count; j++)
                {
                    showlist.Add(wrhlist[i].GoodsLst[j]);
                }
            }
        }

        public static void TwoLevelSort(int lv1id, int lv2id, string sortmod1, string
sortmod2, List<Goods> showlist)
        {
            switch (lv1id)
            {
                case 0:
                    switch (lv2id)
                    {
                        case 1:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenBy(s => s.Categ.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenBy(s => s.Categ.ToString()).ToList();
                            }
```

```
                            showlist = Singleton.showlist;
                            break;
                        case 2:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenByDescending(s => s.Mod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenBy(s => s.Mod.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenByDescending(s => s.Mod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenBy(s => s.Mod.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 3:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenBy(s => s.Prod.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenBy(s => s.Prod.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 4:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenByDescending(s => s.Worth).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenBy(s => s.Worth).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenByDescending(s => s.Worth).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenBy(s => s.Worth).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
```

```
                        case 5:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenBy(s => s.ArDate.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenBy(s => s.ArDate.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 6:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenBy(s => s.InStock.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenBy(s => s.InStock.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 7:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 8:
                            if (sortmod1 == "По убыванию")
```

```
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenByDescending(s => s.WrhId.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Id.ToString()).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenByDescending(s => s.WrhId.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Id.ToString()).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                    }
                    break;

                case 1:
                    switch (lv2id)
                    {
                        case 0:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenByDescending(s => s.Id.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenBy(s => s.Id.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenByDescending(s => s.Id.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenBy(s => s.Id.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 2:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenByDescending(s => s.Mod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenBy(s => s.Mod.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenByDescending(s => s.Mod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenBy(s => s.Mod.ToString()).ToList();
                            }
```

```
                            showlist = Singleton.showlist;
                            break;
                        case 3:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenBy(s => s.Prod.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenBy(s => s.Prod.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 4:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenByDescending(s => s.Worth).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenBy(s => s.Worth).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenByDescending(s => s.Worth).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenBy(s => s.Worth).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 5:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenBy(s => s.ArDate.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenBy(s => s.ArDate.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
```

```
                        case 6:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenBy(s => s.InStock.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenBy(s => s.InStock.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 7:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 8:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenByDescending(s => s.WrhId.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Categ.ToString()).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenByDescending(s => s.WrhId.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Categ.ToString()).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                    }
                    break;
```

```
                case 2:
                    switch (lv2id)
                    {
                        case 1:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenBy(s => s.Categ.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenBy(s => s.Categ.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 0:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenByDescending(s => s.Id.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenBy(s => s.Id.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenByDescending(s => s.Id.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenBy(s => s.Id.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 3:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenBy(s => s.Prod.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenBy(s => s.Prod.ToString()).ToList();
                            }
```

```
                            showlist = Singleton.showlist;
                            break;
                        case 4:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenByDescending(s => s.Worth).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenBy(s => s.Worth).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenByDescending(s => s.Worth).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenBy(s => s.Worth).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 5:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenBy(s => s.ArDate.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenBy(s => s.ArDate.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 6:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenBy(s => s.InStock.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenBy(s => s.InStock.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
```

```
                        case 7:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 8:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenByDescending(s => s.WrhId.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Mod.ToString()).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenByDescending(s => s.WrhId.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Mod.ToString()).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                    }
                    break;

                case 3:
                    switch (lv2id)
                    {
                        case 1:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenBy(s => s.Categ.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
```

```
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenBy(s => s.Categ.ToString()).ToList();
                                }
                                showlist = Singleton.showlist;
                                break;
                            case 2:
                                if (sortmod1 == "По убыванию")
                                {
                                    if (sortmod2 == "По убыванию")
                                        Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenByDescending(s => s.Mod.ToString()).ToList();
                                    else
                                        Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenBy(s => s.Mod.ToString()).ToList();
                                }
                                else
                                {
                                    if (sortmod2 == "По убыванию")
                                        Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenByDescending(s => s.Mod.ToString()).ToList();
                                    else
                                        Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenBy(s => s.Mod.ToString()).ToList();
                                }
                                showlist = Singleton.showlist;
                                break;
                            case 0:
                                if (sortmod1 == "По убыванию")
                                {
                                    if (sortmod2 == "По убыванию")
                                        Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenByDescending(s => s.Id.ToString()).ToList();
                                    else
                                        Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenBy(s => s.Id.ToString()).ToList();
                                }
                                else
                                {
                                    if (sortmod2 == "По убыванию")
                                        Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenByDescending(s => s.Id.ToString()).ToList();
                                    else
                                        Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenBy(s => s.Id.ToString()).ToList();
                                }
                                showlist = Singleton.showlist;
                                break;
                            case 4:
                                if (sortmod1 == "По убыванию")
                                {
                                    if (sortmod2 == "По убыванию")
                                        Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenByDescending(s => s.Worth).ToList();
                                    else
                                        Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenBy(s => s.Worth).ToList();
                                }
                                else
                                {
                                    if (sortmod2 == "По убыванию")
                                        Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenByDescending(s => s.Worth).ToList();
                                    else
                                        Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenBy(s => s.Worth).ToList();
```

```
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 5:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenBy(s => s.ArDate.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenBy(s => s.ArDate.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 6:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenBy(s => s.InStock.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenBy(s => s.InStock.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 7:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
```

```
                            break;
                        case 8:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenByDescending(s => s.WrhId.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Prod.ToString()).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenByDescending(s => s.WrhId.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Prod.ToString()).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                    }
                    break;

                case 4:
                    switch (lv2id)
                    {
                        case 1:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenBy(s => s.Categ.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenBy(s => s.Categ.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 2:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenByDescending(s => s.Mod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenBy(s => s.Mod.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenByDescending(s => s.Mod.ToString()).ToList();
                                else
```

```
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenBy(s => s.Mod.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 3:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenBy(s => s.Prod.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenBy(s => s.Prod.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 0:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenByDescending(s => s.Id.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenBy(s => s.Id.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenByDescending(s => s.Id.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenBy(s => s.Id.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 5:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenBy(s => s.ArDate.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenBy(s => s.ArDate.ToString()).ToList();
```

```
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 6:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenBy(s => s.InStock.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenBy(s => s.InStock.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 7:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 8:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenByDescending(s => s.WrhId.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Worth).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenByDescending(s => s.WrhId.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Worth).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
```

```
                        break;
                    }
                    break;

                case 5:
                    switch (lv2id)
                    {
                        case 1:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenBy(s => s.Categ.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenBy(s => s.Categ.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 2:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenByDescending(s => s.Mod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenBy(s => s.Mod.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenByDescending(s => s.Mod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenBy(s => s.Mod.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 3:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenBy(s => s.Prod.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
```

```
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenBy(s => s.Prod.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 4:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenByDescending(s => s.Worth).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenBy(s => s.Worth).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenByDescending(s => s.Worth).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenBy(s => s.Worth).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 0:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenByDescending(s => s.Id.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenBy(s => s.Id.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenByDescending(s => s.Id.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenBy(s => s.Id.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 6:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenBy(s => s.InStock.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenBy(s => s.InStock.ToString()).ToList();
```

```
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 7:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 8:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenByDescending(s => s.WrhId.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.ArDate.ToString()).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenByDescending(s => s.WrhId.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.ArDate.ToString()).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                    }
                    break;

                case 6:
                    switch (lv2id)
                    {
                        case 1:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenBy(s => s.Categ.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
```

```
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenBy(s => s.Categ.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 2:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenByDescending(s => s.Mod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenBy(s => s.Mod.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenByDescending(s => s.Mod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenBy(s => s.Mod.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 3:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenBy(s => s.Prod.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenBy(s => s.Prod.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 4:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenByDescending(s => s.Worth).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenBy(s => s.Worth).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenByDescending(s => s.Worth).ToList();
```

```
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenBy(s => s.Worth).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 5:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenBy(s => s.ArDate.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenBy(s => s.ArDate.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 0:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenByDescending(s => s.Id.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenBy(s => s.Id.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenByDescending(s => s.Id.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenBy(s => s.Id.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 7:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
```

```
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 8:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenByDescending(s => s.WrhId.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.InStock.ToString()).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenByDescending(s => s.WrhId.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.InStock.ToString()).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                    }
                    break;

                case 7:
                    switch (lv2id)
                    {
                        case 1:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenBy(s => s.Categ.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenBy(s => s.Categ.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 2:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenByDescending(s => s.Mod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenBy(s => s.Mod.ToString()).ToList();
                            }
                            else
                            {
```

```
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenByDescending(s => s.Mod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenBy(s => s.Mod.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 3:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenBy(s => s.Prod.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenBy(s => s.Prod.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 4:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenByDescending(s => s.Worth).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenBy(s => s.Worth).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenByDescending(s => s.Worth).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenBy(s => s.Worth).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 5:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenBy(s => s.ArDate.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
```

```
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenBy(s => s.ArDate.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 6:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenBy(s => s.InStock.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenBy(s => s.InStock.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 0:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenByDescending(s => s.Id.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenBy(s => s.Id.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenByDescending(s => s.Id.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenBy(s => s.Id.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 8:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenByDescending(s => s.WrhId.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.Distr.ToString()).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenByDescending(s => s.WrhId.ToString()).ToList();
```

```
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.Distr.ToString()).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                    }
                    break;

                case 8:
                    switch (lv2id)
                    {
                        case 1:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenBy(s => s.Categ.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenByDescending(s => s.Categ.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenBy(s => s.Categ.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 2:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenByDescending(s => s.Mod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenBy(s => s.Mod.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenByDescending(s => s.Mod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenBy(s => s.Mod.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 3:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenBy(s => s.Prod.ToString()).ToList();
                            }
                            else
```

```
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenByDescending(s => s.Prod.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenBy(s => s.Prod.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 4:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenByDescending(s => s.Worth).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenBy(s => s.Worth).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenByDescending(s => s.Worth).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenBy(s => s.Worth).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 5:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenBy(s => s.ArDate.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenByDescending(s => s.ArDate.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenBy(s => s.ArDate.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 6:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenBy(s => s.InStock.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
```

```
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenByDescending(s => s.InStock.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenBy(s => s.InStock.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 7:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenByDescending(s => s.Distr.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenBy(s => s.Distr.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                        case 8:
                            if (sortmod1 == "По убыванию")
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenByDescending(s => s.Id.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderByDescending(a =>
a.WrhId.ToString()).ThenBy(s => s.Id.ToString()).ToList();
                            }
                            else
                            {
                                if (sortmod2 == "По убыванию")
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenByDescending(s => s.Id.ToString()).ToList();
                                else
                                    Singleton.showlist = showlist.OrderBy(a =>
a.WrhId.ToString()).ThenBy(s => s.WrhId.ToString()).ToList();
                            }
                            showlist = Singleton.showlist;
                            break;
                    }
                    break;
            }
        }

        public static void OneLevelSort(int index, int sort, List<Goods> showlist)
        {
            switch (index)
            {
                case 0:
                    if (sort < 0)
                        showlist.Sort((a, b) => a.Id.CompareTo(b.Id));
                    else showlist.Sort((a, b) => b.Id.CompareTo(a.Id));
                    break;
                case 1:
```

```
                    if (sort < 0)
                        showlist.Sort((a, b) =>
a.Categ.ToString().CompareTo(b.Categ.ToString()));
                    else showlist.Sort((a, b) =>
b.Categ.ToString().CompareTo(a.Categ.ToString()));
                    break;
                case 2:
                    if (sort < 0)
                        showlist.Sort((a, b) =>
a.Mod.ToString().CompareTo(b.Mod.ToString()));
                    else showlist.Sort((a, b) =>
b.Mod.ToString().CompareTo(a.Mod.ToString()));
                    break;
                case 3:
                    if (sort < 0)
                        showlist.Sort((a, b) =>
a.Prod.ToString().CompareTo(b.Prod.ToString()));
                    else showlist.Sort((a, b) =>
b.Prod.ToString().CompareTo(a.Prod.ToString()));
                    break;
                case 4:
                    if (sort < 0)
                        showlist.Sort((a, b) => a.Worth.CompareTo(b.Worth));
                    else showlist.Sort((a, b) => b.Worth.CompareTo(a.Worth));
                    break;
                case 5:
                    if (sort < 0)
                        showlist.Sort((a, b) => a.ArDate.CompareTo(b.ArDate));
                    else showlist.Sort((a, b) => b.ArDate.CompareTo(a.ArDate));
                    break;
                case 6:
                    if (sort < 0)
                        showlist.Sort((a, b) => a.InStock.CompareTo(b.InStock));
                    else showlist.Sort((a, b) => b.InStock.CompareTo(a.InStock));
                    break;
                case 7:
                    if (sort < 0)
                        showlist.Sort((a, b) =>
a.Distr.ToString().CompareTo(b.Distr.ToString()));
                    else showlist.Sort((a, b) =>
b.Distr.ToString().CompareTo(a.Distr.ToString()));
                    break;
                case 8:
                    if (sort < 0)
                        showlist.Sort((a, b) =>
a.WrhId.ToString().CompareTo(b.WrhId.ToString()));
                    else showlist.Sort((a, b) =>
b.WrhId.ToString().CompareTo(a.WrhId.ToString()));
                    break;
            }
        }

        public static void ChangeNew(string id, Category categ, Producer prod, Model mod,
double worth, DateTime ardate, bool instock, Distributor distr)
        {
            //Функция для изменения нового товара (который еще не был добавлен в БД)
            //Принимает на вход идентификатор, категорию, производителя, модель, цену,
дату поступления, наличие и поставщика
            var x = newgoodlist.Find(s => s.Id.ToString() == id);
            x.Categ = categ;
            x.Mod = mod;
            x.Prod = prod;
            x.Worth = worth;
            x.ArDate = ardate;
            x.InStock = instock;
```

```
            x.Distr = distr;
        }

        public static void ChangeGen(string id, Category categ, Producer prod, Model mod,
double worth, DateTime ardate, bool instock, Distributor distr, Warehouse wrh)
        {
            //Функция для изменения существующего товара
            //Принимает на вход идентификатор, категорию, производителя, модель, цену,
дату поступления, наличие, поставщика и номер склада
            var x = wrhlist.Find(s => s == wrh).GoodsLst.Find(u => u.Id == id);
            chgoodlist.Add(new Goods(x.Id, x.Categ, x.Mod, x.Prod, x.Worth, x.ArDate,
x.InStock, x.Distr, x.WrhId));

            x.Categ = categ;
            x.Mod = mod;
            x.Prod = prod;
            x.Worth = worth;
            x.ArDate = ardate;
            x.InStock = instock;
            x.Distr = distr;
        }

        public static void DeleteNew(string id, string wrh)
        {
            //Функция для удаления нового товара
            //Принимает на вход идентификационный номер товара номер его склада
            wrhlist.Find(s => s.ToString() == wrh).GoodsLst.Remove(newgoodlist.Find(s =>
s.Id == id));
            newgoodlist.Remove(newgoodlist.Find(s => s.Id == id));
        }

        public static void DeleteGenNotChanged(string id, string wrh)
        {
            //Функция для удаления существующего товара, который НЕ был предварительно
изменен
            //Принимает на вход идентификационный номер товара номер его склада
            var x = wrhlist.Find(s => s.ToString() == wrh).GoodsLst.Find(s => s.Id ==
id);
            delgoodlist.Add(new Goods(x.Id, x.Categ, x.Mod, x.Prod, x.Worth, x.ArDate,
x.InStock, x.Distr, x.WrhId));
            wrhlist.Find(s => s.ToString() == wrh).GoodsLst.Remove(x);
        }

        public static void DeleteGenChanged(string id, string wrh)
        {
            //Функция для удаления существующего товара, который БЫЛ предварительно
изменен
            //Принимает на вход идентификационный номер товара номер его склада
            var x = chgoodlist.Find(s => s.Id == id);
            delgoodlist.Add(new Goods(x.Id, x.Categ, x.Mod, x.Prod, x.Worth, x.ArDate,
x.InStock, x.Distr, x.WrhId));
            chgoodlist.Remove(x);
            x = wrhlist.Find(s => s.ToString() == wrh).GoodsLst.Find(u => u.Id == id);
            wrhlist.Find(s => s.ToString() == wrh).GoodsLst.Remove(x);
        }

        public static void SaveChanges()
        {
            //Функция для сохранения всех проведенных изменений в БД
            if ((newgoodlist.Count > 0) | (chgoodlist.Count > 0) | (delgoodlist.Count >
0) |
                (newctglist.Count > 0) | (newdislist.Count > 0) | (newprolist.Count > 0)
| (newmodlist.Count > 0))
            {
                DialogResult res = MessageBox.Show(
```

```
                "Внесенные измение невозможно будет отменить. Сохранить изменения?",
                "Сообщение",
                MessageBoxButtons.YesNo,
                MessageBoxIcon.Information,
                MessageBoxDefaultButton.Button1);

                if (res == DialogResult.Yes)
                {
                    string rep;
                    if (newctglist.Count > 0)
                    {
                        //Если были добавлены новые категории, они добавляются в БД
                        for (int i = 0; i < newctglist.Count; i++)
                        {
                            var x = newctglist[i];

                            using (SqlConnection connection = new
SqlConnection(connectionString))
                            {
                                connection.Open();
                                SqlCommand command = new SqlCommand(String.Format("INSERT
INTO Categories (ctg_id, ctg_name) VALUES ('{0}', '{1}')",

x.Id.ToString(), x.Name.ToString()), connection);
                                var a = command.ExecuteNonQuery();
                            }
                            rep = String.Format("{0} - Category added  - {1}",
DateTime.Now, x.ToString());
                            Log(rep);
                        }
                    }
                    if (newdislist.Count > 0)
                    {
                        //Если были добавлены новые поставщики, они добавляются в БД
                        for (int i = 0; i < newdislist.Count; i++)
                        {
                            var x = newdislist[i];

                            using (SqlConnection connection = new
SqlConnection(connectionString))
                            {
                                connection.Open();
                                SqlCommand command = new SqlCommand(String.Format("INSERT
INTO Distributors (distr_id, distr_name) VALUES ('{0}', '{1}')",

x.Id.ToString(), x.Name.ToString()), connection);
                                var a = command.ExecuteNonQuery();
                            }
                            rep = String.Format("{0} - Distributor added  - {1}",
DateTime.Now, x.ToString());
                            Log(rep);
                        }
                    }
                    if (newprolist.Count > 0)
                    {
                        //Если были добавлены новые производители, они добавляются в БД
                        for (int i = 0; i < newprolist.Count; i++)
                        {
                            var x = newprolist[i];

                            using (SqlConnection connection = new
SqlConnection(connectionString))
                            {
                                connection.Open();
```

```
                                SqlCommand command = new SqlCommand(String.Format("INSERT
INTO Producers (prod_id, prod_name) VALUES ('{0}', '{1}')",

x.Id.ToString(), x.Name.ToString()), connection);
                                var a = command.ExecuteNonQuery();
                            }
                            rep = String.Format("{0} - Producer added  - {1}",
DateTime.Now, x.ToString());
                            Log(rep);
                        }
                    }
                    if (newmodlist.Count > 0)
                    {
                        //Если были добавлены новые модели, они добавляются в БД
                        for (int i = 0; i < newmodlist.Count; i++)
                        {
                            var x = newmodlist[i];

                            using (SqlConnection connection = new
SqlConnection(connectionString))
                            {
                                connection.Open();
                                SqlCommand command = new SqlCommand(String.Format("INSERT
INTO Models (mod_id, mod_name, ctg_id, prod_id) VALUES ('{0}', '{1}', '{2}', '{3}')",

x.Id.ToString(), x.Name.ToString(), x.Categ.Id.ToString(), x.Prod.Id.ToString()),
connection);
                                var a = command.ExecuteNonQuery();
                            }
                            rep = String.Format("{0} - Model added  - {1}", DateTime.Now,
x.ToString());
                            Log(rep);
                        }
                    }
                    if (newgoodlist.Count > 0)
                    {
                        //Если были добавлены новые товары, они добавляются в БД
                        for (int i = 0; i < newgoodlist.Count; i++)
                        {
                            var x = newgoodlist[i];

                            using (SqlConnection connection = new
SqlConnection(connectionString))
                            {
                                connection.Open();
                                SqlCommand command = new SqlCommand(String.Format("INSERT
INTO Goods (good_id, mod_id, worth, arrive_date, in_stock, distr_id, wrh_id) VALUES
('{0}', '{1}', {2}, '{3}', {4}, '{5}', '{6}')",

x.Id.ToString(), x.Mod.Id.ToString(), x.Worth.ToString(), x.ArDate.ToString("yyyy.MM.dd
HH:mm:ss"), (Convert.ToInt32(x.InStock)).ToString(), x.Distr.Id.ToString(),
x.WrhId.Id.ToString()), connection);
                                var a = command.ExecuteNonQuery();
                            }
                            rep = String.Format("{0} - Warehouse {1} - Good added  -
{2}", DateTime.Now, x.WrhId.ToString(), x.ToString());
                            Log(rep);
                        }
                    }
                    if (chgoodlist.Count > 0)
                    {
                        //Если товары были изменены, изменения вносятся в БД
                        for (int i = 0; i < chgoodlist.Count; i++)
                        {
```

```
                            var x = wrhlist.Find(s => s ==
chgoodlist[i].WrhId).GoodsLst.Find(u => u.Id == chgoodlist[i].Id);

                            using (SqlConnection connection = new
SqlConnection(connectionString))
                            {
                                connection.Open();
                                SqlCommand command = new SqlCommand(String.Format("UPDATE
Goods SET mod_id='{1}', worth={2}, arrive_date='{3}', in_stock={4}, distr_id='{5}' WHERE
good_id='{0}'",

x.Id.ToString(), x.Mod.Id.ToString(), x.Worth.ToString(), x.ArDate.ToString("yyyy.MM.dd
HH:mm:ss"), (Convert.ToInt32(x.InStock)).ToString(), x.Distr.Id.ToString()), connection);
                                var a = command.ExecuteNonQuery();
                            }
                            rep = String.Format("{0} - Warehouse {1} - Good changed -
{2}", DateTime.Now, x.WrhId.ToString(), x.ToString());
                            Log(rep);
                        }
                    }
                    if (delgoodlist.Count > 0)
                    {
                        //Если товары были удалены, эти товары удаляются из БД
                        for (int i = 0; i < delgoodlist.Count; i++)
                        {
                            var x = delgoodlist[i];

                            using (SqlConnection connection = new
SqlConnection(connectionString))
                            {
                                connection.Open();
                                SqlCommand command = new SqlCommand(String.Format("DELETE
FROM Goods WHERE good_id='{0}'",

x.Id.ToString()), connection);
                                var a = command.ExecuteNonQuery();
                            }
                            rep = String.Format("{0} - Warehouse {1} - Good deleted -
{2}", DateTime.Now, x.WrhId.ToString(), x.ToString());
                            Log(rep);
                        }
                    }
                    MessageBox.Show(
                        String.Format("Добавлено записей: {0}\nИзменено записей:
{1}\nУдалено записей: {2}",
                        newgoodlist.Count + newmodlist.Count + newctglist.Count +
newdislist.Count + newprolist.Count, chgoodlist.Count, delgoodlist.Count),
                        "Сообщение",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Information,
                        MessageBoxDefaultButton.Button1);

                    newgoodlist.Clear();
                    chgoodlist.Clear();
                    delgoodlist.Clear();
                    newctglist.Clear();
                    newdislist.Clear();
                    newprolist.Clear();
                    newmodlist.Clear();
                }
            }
            else
                MessageBox.Show(
                    "Изменений не внесено",
                    "Сообщение",
```

```
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information,
                    MessageBoxDefaultButton.Button1);
        }

        public static void CancelChanges()
        {
            //Функция для отмены всех проведенных изменений
            DialogResult res = MessageBox.Show(
                "Несохраненные измениния будут потеряны. Отменить изменения?",
                "Сообщение",
                MessageBoxButtons.YesNo,
                MessageBoxIcon.Information,
                MessageBoxDefaultButton.Button1);

            if (res == DialogResult.Yes)
            {
                //Удаление всех новых товаров из списков
                for (int i = 0; i < wrhlist.Count; i++)
                {
                    for (int j = 0; j < wrhlist[i].GoodsLst.Count; j++)
                    {
                        if (newgoodlist.Find(s => wrhlist[i].GoodsLst[j].Id == s.Id) ==
wrhlist[i].GoodsLst[j])
                        {
                            wrhlist[i].GoodsLst.RemoveAt(j);
                            j -= 1;
                        }
                    }
                }
                //Возвращение измененных товаров к первоначальному виду
                for (int i = 0; i < chgoodlist.Count; i++)
                {
                    var x = wrhlist.Find(s => s.Id ==
chgoodlist[i].WrhId.Id).GoodsLst.Find(u => u.Id == chgoodlist[i].Id);
                    wrhlist.Find(s => s.Id == chgoodlist[i].WrhId.Id).GoodsLst.Remove(x);
                    x = chgoodlist[i];
                    wrhlist.Find(s => s.Id == chgoodlist[i].WrhId.Id).GoodsLst.Add(new
Goods(x.Id, x.Categ, x.Mod, x.Prod, x.Worth, x.ArDate, x.InStock, x.Distr, x.WrhId));
                }
                //Возвращение в списки удаленных товаров
                for (int i = 0; i < delgoodlist.Count; i++)
                {
                    var x = delgoodlist[i];
                    wrhlist.Find(s => s.Id == delgoodlist[i].WrhId.Id).GoodsLst.Add(new
Goods(x.Id, x.Categ, x.Mod, x.Prod, x.Worth, x.ArDate, x.InStock, x.Distr, x.WrhId));
                }
                //Удаление всех новых моделей из списков
                for (int i = 0; i < newmodlist.Count; i++)
                {
                    modlist.Remove(modlist.Find(s => s.Id == newmodlist[i].Id));
                }
                //Удаление всех новых категорий из списков
                for (int i = 0; i < newctglist.Count; i++)
                {
                    ctglist.Remove(ctglist.Find(s => s.Id == newctglist[i].Id));
                }
                //Удаление всех новых поставщиков из списков
                for (int i = 0; i < newdislist.Count; i++)
                {
                    dislist.Remove(dislist.Find(s => s.Id == newdislist[i].Id));
                }
                //Удаление всех новых производителей из списков
                for (int i = 0; i < newprolist.Count; i++)
                {
```

```
                    prolist.Remove(prolist.Find(s => s.Id == newprolist[i].Id));
                }

                newgoodlist.Clear();
                chgoodlist.Clear();
                delgoodlist.Clear();
                newctglist.Clear();
                newdislist.Clear();
                newprolist.Clear();
                newmodlist.Clear();
            }
        }

        public static void Exit()
        {
            //Функция закрытия приложения
            DialogResult res = MessageBox.Show(
                "Несохраненные изменения будут потеряны. Закрыть проект?",
                "Сообщение",
                MessageBoxButtons.YesNo,
                MessageBoxIcon.Information,
                MessageBoxDefaultButton.Button1);

            if (res == DialogResult.Yes)
                Application.Exit();
        }

        public static void Log(string message)
        {
            File.AppendAllText("log.txt", message + "\n");
        }
    }
}
```