СОДЕРЖАНИЕ

ВВЕДЕНИЕ....2
1. Выбор объекта автоматизации и описание его бизнес-процессов....3
2. Объектно-ориентированный анализ предметной области....6
3. Проектирование....11
4. Тесты....167
5. Пример работы программы....190
ЗАКЛЮЧЕНИЕ....201
СПИСОК ЛИТЕРАТУРЫ....22
ПРИЛОЖЕНИЕ А....23

ВВЕДЕНИЕ

Целью курсовой работы по дисциплинам «Программирование», «Объектно-ориентированное программирование» является закрепление и углубление знаний, полученных при изучении дисциплины в ходе лекционных и практических занятий, получение практических навыков создания программ с использованием объектно-ориентированной технологии в современных средах проектирования.

Задание для курсовой работы – разработка системы автоматизации учета в пиццерии

Выполнение курсовой работы предусматривает решение следующих **задач**:

1) Выбрать объект автоматизации и описать его бизнес-процессы;
2) Выполнить объектно-ориентированный анализ предметной области;
3) Разработать диаграмму классов;
4) Разработать сценарии диалога с пользователем и диалоговые окна;
5) Выбрать структуры данных (коллекции для хранения информации, использовать стандартные коллекции C#);
6) Реализовать ввод данных в коллекции (с клавиатуры и автоматический);
7) Реализовать сохранение данных в файл и загрузку из файла (использовать сериализацию и диалоговые окна для выбора файла);
8) Реализовать сохранение информации о выполненных действиях в журнал (использовать события);
9) Реализовать конструктор запросов (использовать LINQ/методы расширения).

1. Выбор объекта автоматизации и описание его бизнес-процессов

Объектом автоматизации выбрана пиццерия. Описание бизнес-процессов приведено в таблице 1.

Таблица 1 – Описание бизнес-процессов пиццерии

Действующее лицо	Бизнес-процесс	Содержание бизнес-процесса	Артефакты
Кассир	Прием заказа у клиента	1)Кассир формирует заказ из позиций, указанных в меню пиццерии, исходя из пожеланий клиента. 2) Кассир отправляет сформированный заказ в систему для его выполнения.	Заказ
Повар	Приготовление блюд в заказе	При достаточном количестве продуктов на складе, повар может приготовить блюда в заказе. Если продуктов недостаточно для приготовления всех блюд, повар отправляет запрос на склад для покупки недостающих продуктов	Заказ, продукты на складе
Курьер	Доставка заказа	Если для выполнения заказа требуется его доставка, курьер	Заказ

		выполняет доставку заказа клиенту. Если доставка не требуется, выполненный заказ сразу отдается клиенту	
Директор пиццерии, работники пиццерии	Начисление з/п сотрудникам по результатам работы	1) Директор рассчитывает зарплату работникам, исходя из установленной для пиццерии ставки для определенной должности 2) Директор назначает зарплату работникам	
Директор пиццерии, работники пиццерии	Принятие на работу новых работников	Директор автосервиса заносит в список нового работника	
Директор пиццерии	Закупка продуктов	Директор закупает недостающие продукты на склад, руководствуясь списком запрошенных продуктов	
Директор пиццерии	Формирование или изменение меню	Директор добавляет или удаляет позиции меню	Меню

2. Объектно-ориентированный анализ предметной области

Объектно-ориентированный анализ предметной области представлен в таблице 2.1

Таблица 2.1 – Объектно-ориентированный анализ предметной области. Классы.

Класс	Назначение	Реализация	Действия
Pizzery	Пиццерия	Класс	CRUD*
Staff	Штат сотрудников	Класс	CRUD
Worker	Работник	Класс	CRUD
Storage	Склад	Класс	CRUD
Product	Продукт	Класс	CRUD
Ingredient	Ингредиент для пиццы	Класс, производный от Product	CRUD
Menu	Меню пиццерии	Класс	CRUD
MenuPosition	Позиция меню	Класс	CRUD
Pizza	Пицца	Класс, производный от MenuPosition	CRUD
Drink	Напиток	Класс, производный от MenuPosition	CRUD
OtherFood	Другое блюдо	Класс, производный от MenuPosition	CRUD
OrderList	Список заказов	Класс	CRUD
Order	Заказ	Класс	CRUD
Wallet	Счет пиццерии	Класс	CRUD

*C (create) – создать, R (read) – просмотреть, U (update) – обновить/изменить, D (delete) – удалить.

Анализ полей классов представлен в таблице 2.2.

Таблица 2.2 –Анализ полей классов. Атрибуты.

Имя класса	Поле	Тип	Назначение	Ограничения
Pizzery	staff	Staff	Штат сотрудников	Объект класса Staff
	storage	Storage	Склад пиццерии	Объект класса Storage
	menu	Menu	Меню пиццерии	Объект класса Menu
	orderList		Список заказов	Объект класса OrderList
	wallet		Счет пиццерии	Объект класса Wallet
Staff	Workers	List<Worker>	Список сотрудников пиццерии	Список объектов класса Worker
	count	int	Порядковый номер последнего добавленного сотрудника	Целое число, больше или равное 0
	Count	int	Количество сотрудников	Свойство, только для чтения, возвращает количество сотрудников

Worker	Name	string	Ф,И,О сотрудника	Стока, содержит буквы.
	Position	string	Должность сотрудника	Стока, содержит буквы.
	OrderCount	int	Количество обработанных заказов	Целое число, больше или равное 0
	Index	int	Индекс сотрудника в штате	Целое число, больше или равное 0
Storage	Content	List<Product>	Список продуктов на складе	Список объектов класса Product
	ProductQuery	List<Product>	Список запрошенных продуктов	Список объектов класса Product
Product	Name	string	Список из деталей	Стока, содержит буквы.
	Cost	double	Стоимость продукта	Число с плавающей точкой большее или равное 0
	CountOnStorage	int	Количество продукта	Целое число, больше или равное 0
Menu	Content	List<MenuPosition>	Содержание меню	Список содержит

				объекты типа MenuPosition
	deliveryCost	int	Стоимость доставки заказа	Целое число, больше или равное 0
OrderList	orders	List<Order>	Список заказов пиццерии	Список содержит объекты типа Order
Order	Cost	double	Общая стоимость всех позиций заказа	Число с плавающей точкой большее или равное 0
	Content	List<MenuPosition>	Содержание заказа	Список содержит объекты типа MenuPosition
	WorkersInvolved	List<Worker>	Список сотрудников, принимавших участие в обработке заказа	Список содержит объекты типа Worker
	Status	string	Статус выполнения заказа	Стока, содержит буквы.
	delivery	bool	Нужна ли доставка для данного заказа	Значения true или false
	deliveryCost;	double	Стоимость доставки данного заказа	Число с плавающей точкой большее или равное 0

	Index	int	Индекс заказа	Целое число, больше или равное 0
MenuPosition	Name	string	Наименование позиции	Стока, содержит буквы.
	Cost	double	Стоимость позиции	Число с плавающей точкой >=0
	Index	int	Индекс позиции	Целое число, больше или равное 0
Pizza	Ingredients	List<Ingredient>	Список ингредиентов пиццы	Список содержит объекты типа Ingredient
Drink	productInStorage	Product	Продукт на складе, соответствующий упаковке напитка	Объект типа Product
OtherFood	Ingredients	List<Ingredient>	Список ингредиентов блюда	Список содержит объекты типа Ingredient
Wallet	incoming	double	Доходы пиццерии	Целое число, больше или равное 0
	spending	double	Расходы пиццерии	Целое число, больше или равное 0
	salaries	int[]	Зарплаты для должностей	Массив целых чисел больших или равных 0

3. Проектирование

а) Диаграмма классов представлена на рисунке 3.1

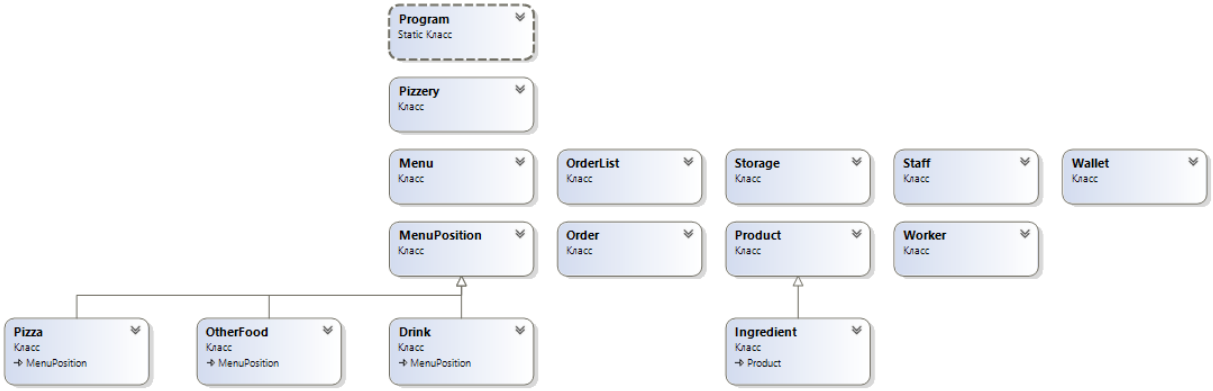

Рисунок 3.1 – Диаграмма классов

Дерево форм представлено на рисунке 3.2

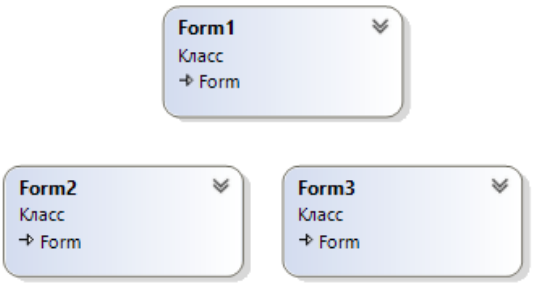

Рисунок 3.2 – Дерево форм

б) Внешний вид форм представлен на рисунках 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 3.10.

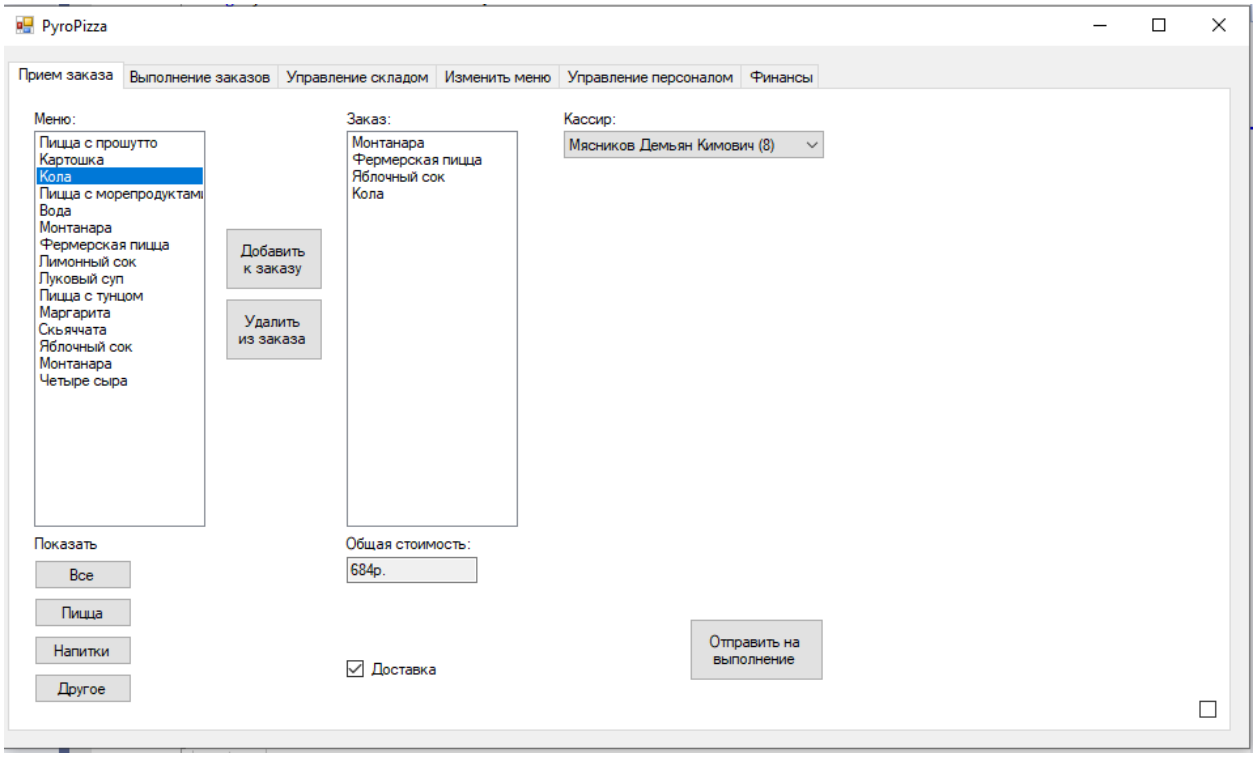

Рисунок 3.3 – Form1: Прием заказа

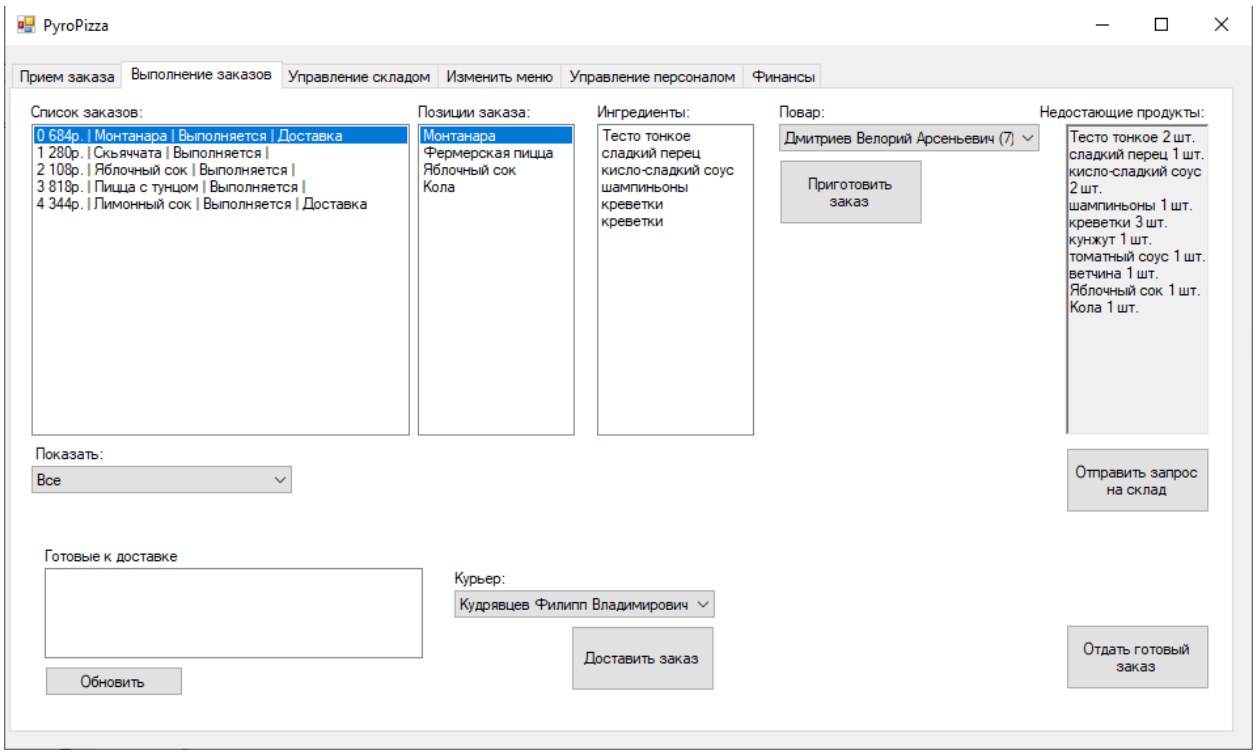

Рисунок 3.4 –Form1: Выполнение заказа

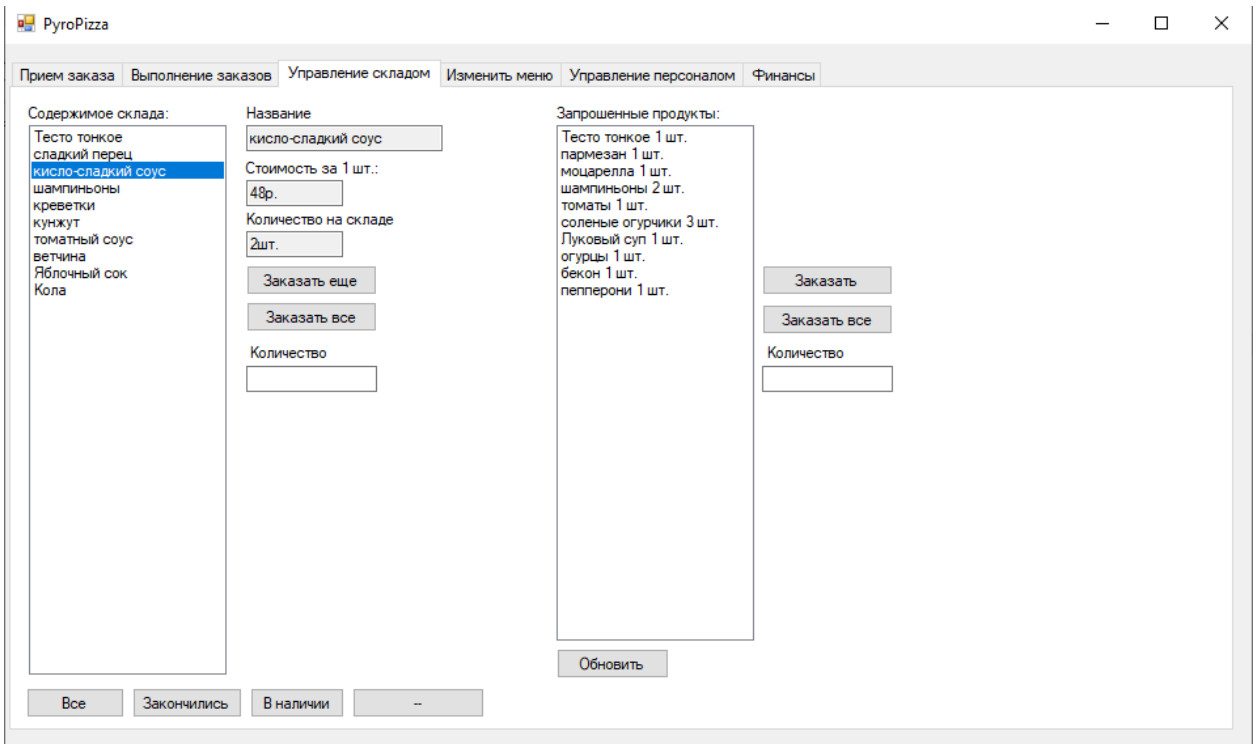

Рисунок 3.5 – Form1: Управление складом

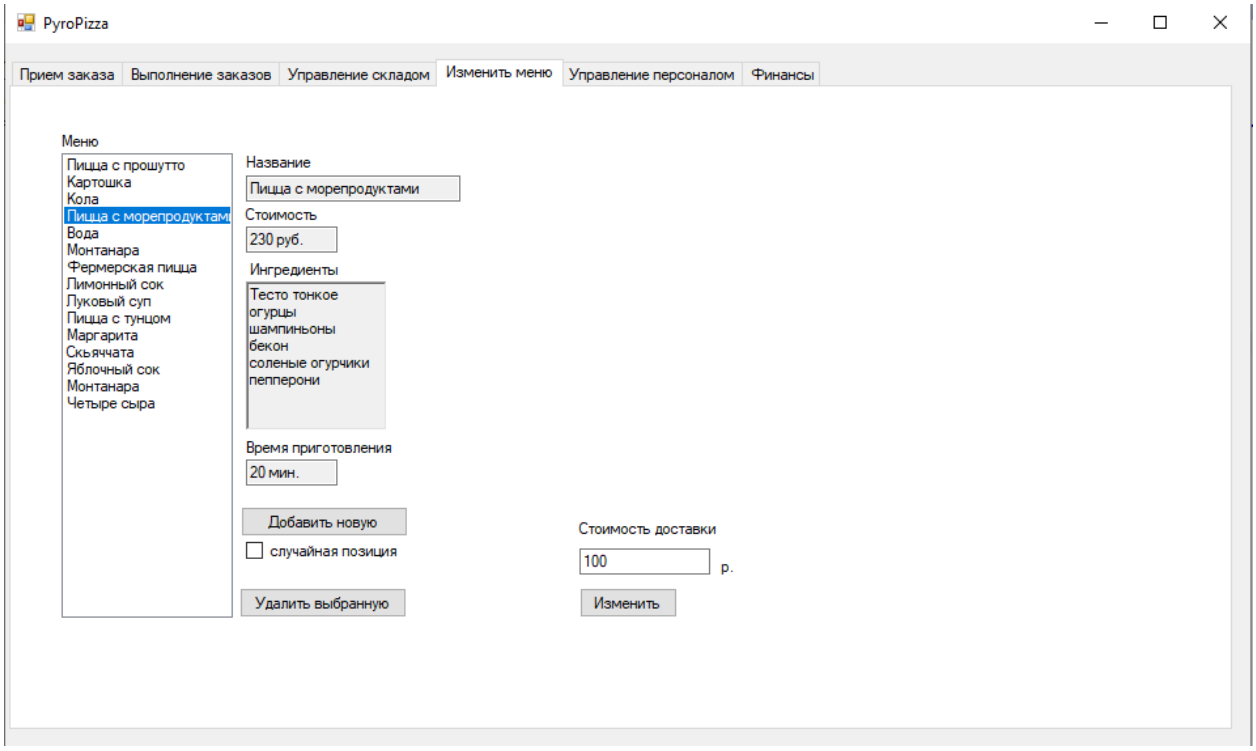

Рисунок 3.6 –Form1: Изменение меню

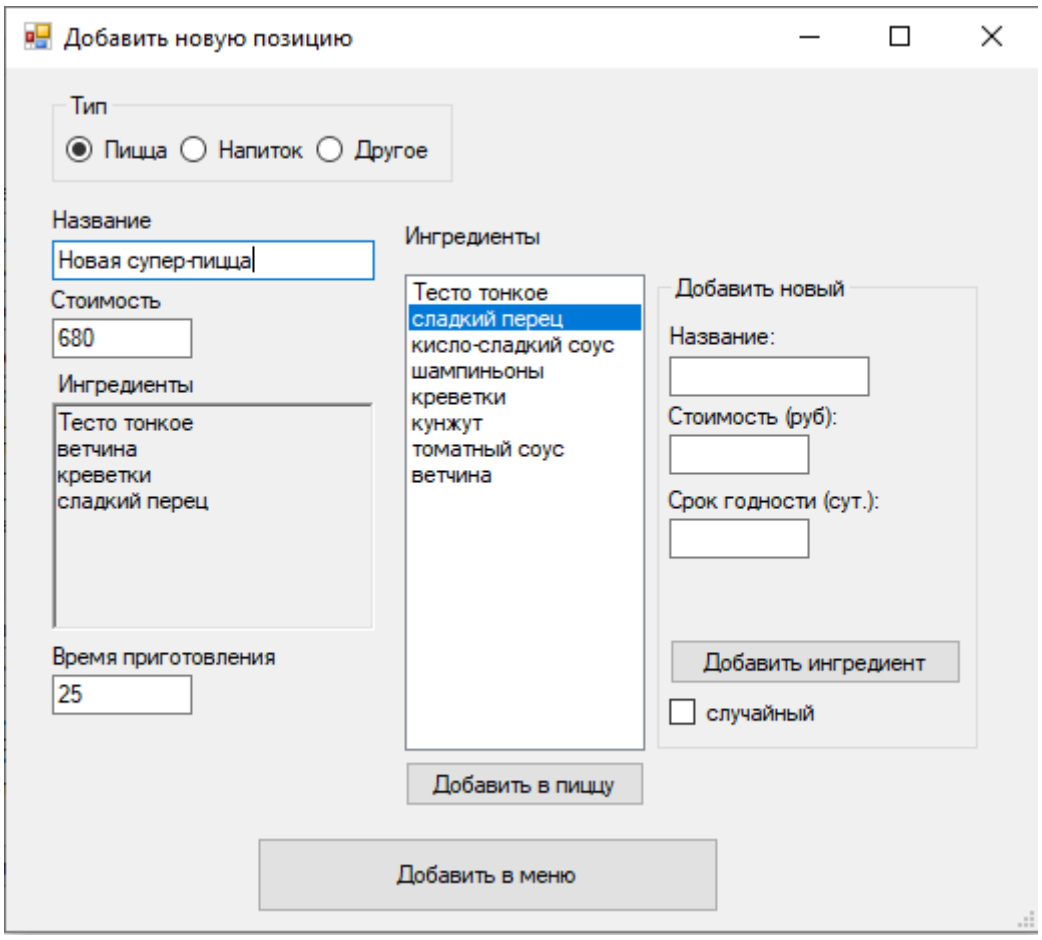

Рисунок 3.7 – Form2: Добавление новой позиции в меню

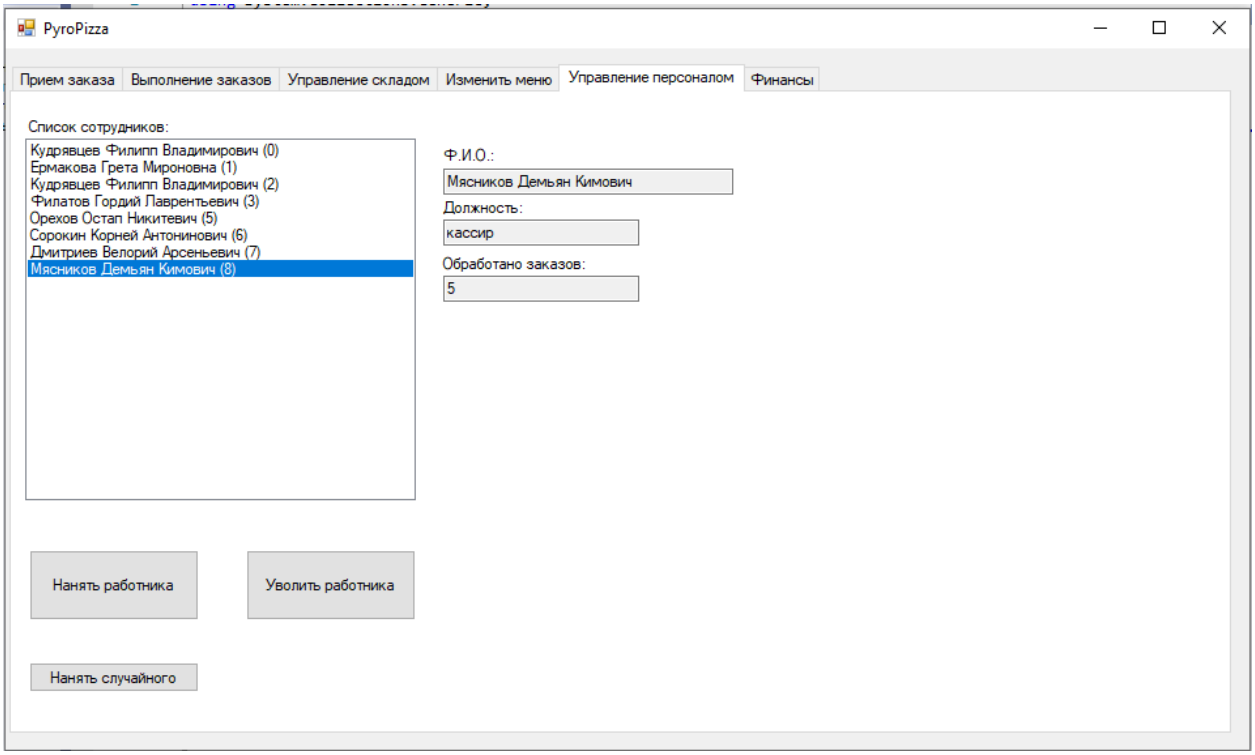

Рисунок 3.8 – Form1: Управление персоналом

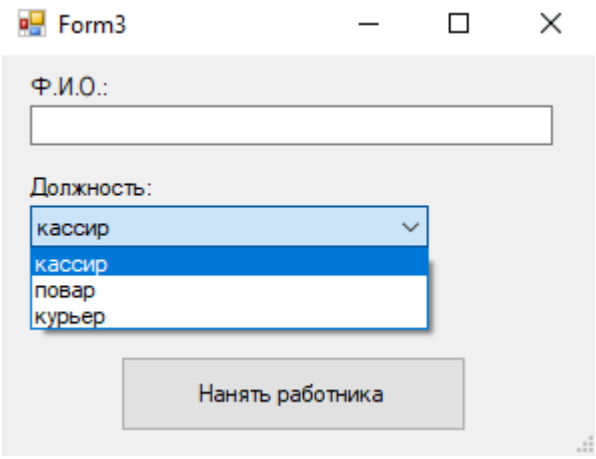

Рисунок 3.9 – Form3: Добавление нового работника

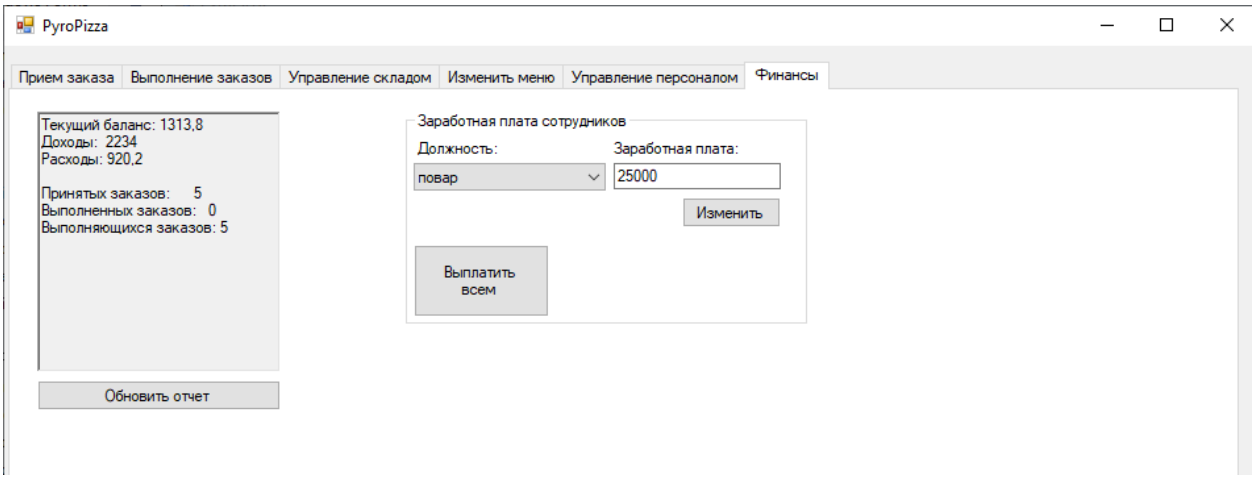

Рисунок 3.10 – Form1: Финансовый отчет и выплата зарплат

Описание форм представлено в таблице 3.1

Таблица 3.1 – Описание форм

Форма	Описание
Form1	Стартовый экран программы. Первая вкладка формы позволяет сформировать заказ, принятый от клиента и отправить его на выполнение. Вторая вкладка осуществляет выполнение стадий обработки заказа, а именно приготовление блюд и доставку. Третья вкладка нужна для покупки недостающих продуктов на склад пиццерии. Четвертая вкладка осуществляет добавление и удаление позиций в меню пиццерии. Пятая вкладка отображает список сотрудниов и позволяет добавить или удалить запись о сотруднике. Шестая вкладка отображает финансовый отчет пиццерии и позволяет начислить зарплату сотрудникам.
Form2	Форма для добавления новой позиции в меню пиццерии.
Form3	Форма для добавления нового сотрудника.

4. Тесты

В таблице 4.1 представлены тесты работы программы.

Таблица 4.1 – Тесты работы программы

№	Действие	Реакция системы	Результат
1	Запуск программы	Открывается окно «PyroPizza» с выбранной вкладкой «Прием заказа»	Верно
2	Клик ЛКМ на пункт в списке «Меню» затем клик ЛКМ на кнопку «Добавить к заказу»	В список «Заказ» добавляется выбранный пункт	Верно
3	Клик ЛКМ на кнопку «Пицца» под подписью «Показать»	В списке «Меню» отображаются только пункты, обозначающие пиццу	Верно
4	Клик ЛКМ на выпадающий список «Кассир»	Отображается список нанятых кассиров в пиццерии	Верно
5	Клик ЛКМ на кнопку «Отправить на выполнение»	Очищается список «Заказ», заказ добавляется в список заказов на вкладке2	Верно
6	Клик ЛКМ на пункт из списка «Заказ» ЛКМ на кнопку «Удалить из заказа»	Выбранный пункт удаляется из списка	Верно
7	Клик ЛКМ на вкладку «Выполнение заказов»	Открывается вкладка «Отправление на ремонт» со списком машин в очереди на ремонт	Верно
8	Клик ЛКМ на заказ из списка «Список заказов»	Заполняется список «Позиции заказа», а также «Недостающие продукты»	Верно
9	Клик ЛКМ на позицию из списка «Позиции заказа»	В списке «Ингредиенты» появились ингредиенты,	Верно

		необходимые приготовления позиции	
10	Клик ЛКМ на кнопку «Приготовить заказ»	Сообщение об ошибке «Для приготовления заказа нужен повар»	Верно
11	Клик ЛКМ на выпадающий список «Повар»	Отображается список нанятых поваров в пиццерии	Верно
12	Клик ЛКМ на кнопку «Приготовить заказ»	Сообщение об ошибке «Недостаточно продуктов на складе»	Верно
13	Клик ЛКМ на кнопку «Отправить запрос на склад»	Сообщение «Запрос на склад отправлен», недостающие продукты добавлены в очередь склада	Верно
14	Клик ЛКМ на вкладку «Выполнение заказов»	Открывается вкладка «Управление складом» со списками «Содержимое склада» и «Запрошенные продукты»	Верно
15	Клик ЛКМ на кнопку «Заказать все» рядом со списком «Запрошенные продукты»	Все продукты из списка добавляются на склад в необходимом количестве	Верно
16	Клик ЛКМ на пункт из списка «Содержимое склада»	Вывод информации о продукте в соответствующие текстовые поля	Верно
17	Клик ЛКМ на вкладку «Изменить меню»	Открывается вкладка «Изменить меню» со списком позиций	Верно
18	Поставить галочку «Случайная позиция» и ЛКМ на кнопку «Добавить новую»	В список позиций добавляется новая случайная позиция	Верно
19	Снять галочку «Случайная позиция» и	Открывается окно «Добавить новую позицию»	Верно

	ЛКМ на кнопку «Добавить новую»		
20	Ввод информации о позиции, добавление ингредиентов из списка доступных и ЛКМ по кнопке «Добавить в меню»	Окно закрывается, в списке «Меню» появляется новая позиция	Верно
21	Клик ЛКМ на позицию из списка	В соответствующие поля выводится информация о выбранной позиции	Верно
22	Клик ЛКМ на кнопку «Удалить выбранную»	Выбранная позиция удаляется из списка	Верно
23	Клик ЛКМ на вкладку «Управление персоналом»	Открывается вкладка «Управление персоналом» со списком всех нанятых сотрудников	Верно
24	Клик ЛКМ на пункт из списка «Список сотрудников»	В соответствующие поля выводится информация о выбранном сотруднике	Верно
25	Клик ЛКМ кнопку «Нанять случайного»	В списке появляется новый случайный сотрудник	Верно
26	Клик ЛКМ кнопку «Нанять сотрудника»	Открывается окно «Добавить сотрудника»	Верно
27	Заполнение полей и ЛКМ на кнопку «Нанять работника»	В список сотрудников добавляется новая запись с указанными данными	Верно
28	Клик ЛКМ кнопку «Уволить работника»	Из списка удаляется выбранный сотрудник	Верно

5. Пример работы программы

После запуска программы появляется главное окно, представленное на рисунке 3.3. Если пользователь уже работал с программой, данные автоматически восстановятся после последней сессии работы. На главном окне присутствует несколько вкладок для работы с пиццерией, (рисунок 3.3, 3.4, 3.5, 3.6, 3.8, 3.10). Выбранным кассиром формируется заказ и отправляется на обработку. Во вкладке «Выполнение заказов» Выбирается заказ из списка доступных и выполняется его приготовление выбранным поваром. Если на складе недостаточно продуктов, можно отправить запрос о покупке недостающих продуктов. Недостающие продукты могут быть закуплены сразу на склад во вкладке «Управление складом». После приготовления всех позиций в заказе, он должен быть доставлен выбранным курьером если для данного заказа требуется доставка. В ином случае, заказ сразу отдается клиенту.

Для управления списком сотрудников предназначена вкладка «Управление персоналом». На ней можно добавить или удалить записи в списке сотрудников.

На вкладке «Финансы» расположена информация о доходах и расходах пиццерии, а также возможность начислять зарплату сотрудникам.

ЗАКЛЮЧЕНИЕ

Реализована программа, которая моделирует процессы в пиццерии. В программе хранятся сведения о продуктах, элементах меню пиццерии, работниках и продуктах. Пользователь может формировать заказы от имени выбранного кассира, вносить изменения в меню пиццерии, собирать заказ из продуктов на складе, а также отслеживать влияние проводимых операций на общий счет пиццерии. Для удобства работы пользователя предусмотрена автоматическая загрузка данных, с которыми работали ранее, а также автоматическое сохранение по закрытию программы.

Поставленная цель была достигнута. Были описаны бизнес-процессы пиццерии, проведен объектно-ориентированный анализ предметной области, разработана диаграмма классов, сценарии диалога с пользователем. Реализовано сохранение и загрузка результатов работы. Выбраны структуры данных, реализован ввод данных с клавиатуры и с помощью ДСЧ.

СПИСОК ЛИТЕРАТУРЫ

1) METANIT.COM Сайт о программировании [Электронный ресурс]. URL: https://metanit.com/sharp/
2) Павловская Т. А. C#. Программирование на языке высокого уровня: Учебник для вузов. - СПб.: БХВ-Петербург. 2007.
3) Нортроп Тони, Уилдермьюс Шон, Райан Билл. Основы разработки приложений на платформе Microsoft .Net Framework. Учебный курс Microsoft / Пер. с англ. - М.: «Русская редакция», СПб.: «Питер», 2007.
4) Вагнер, Билл C# Эффективное программирование / Билл Вагнер. - М.: ЛОРИ, 2013. - 320 с.

ПРИЛОЖЕНИЕ А

```
using System;
using System.Collections.Generic;

namespace PyroPizza
{
    [Serializable]
    class Pizzery
    {
        public Staff staff;
        public Storage storage;
        public Menu menu;
        public Journal journal;
        public OrderList orderList;
        public Wallet wallet;
        public Pizzery()
        {
            staff = new Staff();
            storage = new Storage();
            menu = new Menu();
            journal = new Journal();
            orderList = new OrderList();
            wallet = new Wallet();
        }
        public List<Ingredient> GetKnownIngredients()
        {
            List<Ingredient> res = new List<Ingredient>();

            foreach (var i in storage.Content)
            {
                Ingredient ai = i as Ingredient;
                if (ai != null)
                    res.Add(ai);
            }
            return res;
        }
        public void SetSalary(int ind, int sal)
        {
            wallet.SetSalary(ind, sal);
        }
        public void DismissWorker(int ind)
        {
            staff.DismissWorker(ind);
        }
        public void BuyAllStorage(int count)
        {
            wallet.Spend(storage.AppendAll(count));
        }
        public void PayToWorkers()
        {
            if (staff.Count == 0) throw new ArgumentException("Некому платить");

            double cost = 0;
            foreach (var i in staff.Workers)
            {
                if (i.Position == "кассир")
                    cost += wallet.GetSalary(0);
                else if(i.Position == "повар")
                    cost += wallet.GetSalary(1);
                else if (i.Position == "курьер")
                    cost += wallet.GetSalary(2);
            }
            wallet.Spend(cost);
```

```
        }
        public void BuyRequestedOnStorage(int count)
        {
            wallet.Spend(storage.AppendAllQeue(count));
        }
        public void BuyRequestedOnStorage()
        {
            wallet.Spend(storage.AppendAllQeue());
        }
        public void BuyProductOnStorage(int cnt)
        {
            wallet.Spend(storage.AppendAllContent(cnt));
        }
        public void BuyProductOnStorage(Product p, int cnt)
        {
            wallet.Spend(storage.AppendPosition(p, cnt));
        }
        public void AddWorker()
        {
            staff.Add(new Worker());
        }
        public void AddWorker(Worker w)
        {
            staff.Add(w);
        }
        public void AcceptOrder(Order ord)
        {
            orderList.Add(ord);
            wallet.Income(ord.Cost);
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;

namespace PyroPizza
{
    [Serializable]
    class Storage
    {
        public List<Product> Content { get; }
        public List<Product> ProductQuery { get { return productQuery; } }
        private List<Product> productQuery;
        public Storage()
        {
            Content = new List<Product>();
            productQuery = new List<Product>();
        }
        public void Request(List<Product> list)
        {
            foreach (var i in list)
            {
                if (!ProductQuery.Contains(i))
                    productQuery.Add(i);
                else
                {
                    Product ai = productQuery.Find(p => p.Name == i.Name);
                    ai.SetCount( ai.CountOnStorage+ i.CountOnStorage);
                }
            }
        }
        public void PrepareOrd(Order o)
```

```
        {
            if (GetMissingProgucts(o).Count != 0) throw new
ArgumentException("Недостаточно продуктов на складе");

            List<Product> ls = GetProducts(o);

            foreach (var i in ls)
            {
                Product tmp = Content.Find(p => p.Name == i.Name);
                tmp.Spend(1);
            }
        }
        public void Add(Product product)
        {
            Product ai = Content.Find(t => t == product);
            if (ai == null)
            {
                Content.Add(product);
            }
            else
            {
                ai.Append(product.CountOnStorage);
            }
        }
        private void AppQ(Product p, List<Product> ls)
        {
            Product tmp = ls.Find(pp => p.Name == pp.Name);
            if (tmp == null)
            {
                p.SetCount(1);
                ls.Add(p);
            }
            else tmp.Append(1);
        }

        public List<Product> GetProducts(Order ord)
        {
            List<Product> res = new List<Product>();
            foreach (var pos in ord.Content)
            {
                if (pos is Pizza)
                {
                    Pizza pp = pos as Pizza;
                    foreach (var i in pp.Ingredients)
                    {
                        AppQ((Product)i, res);
                    }
                }
                else if (pos is OtherFood)
                {
                    OtherFood po = pos as OtherFood;
                    foreach (var i in po.Ingredients)
                    {
                        AppQ((Product)i, res);
                    }
                }
                else
                {
                    Drink d = pos as Drink;
                    AppQ((Product)d.productInStorage, res);
                }
            }
            return res;
        }
```

```
        public List<Product> GetMissingProgucts(Order ord)
        {
            List<Product> res = new List<Product>();
            List<Product> tmp = GetProducts(ord);
            foreach (var i in tmp)
            {
                Product p = Content.Find(pp => pp.Name == i.Name);
                if (p == null)
                {
                    res.Add(i);
                }
                else
                {
                    if (i.CountOnStorage > p.CountOnStorage)
                    {
                        i.SetCount(i.CountOnStorage - p.CountOnStorage);
                        res.Add(i);
                    }
                }
            }
            return res;
        }
        public void Append(Product prod, int co)
        {
            Product tmp = Content.Find(p => p.Name == prod.Name);
            if (tmp == null)
            {
                int t = prod.CountOnStorage;
                prod.SetCount(co);
                Ingredient tpi = prod as Ingredient;
                if (tpi != null)
                    Content.Add((Ingredient)tpi.Clone());
                else
                    Content.Add((Product)prod.Clone());
                prod.SetCount(t - co);

            }
            else
            {
                tmp.Append(co);
                prod.SetCount(prod.CountOnStorage - co);
            }

        }

        public void SpendByOrder(Order ord)
        {
            List<Product> check = GetMissingProgucts(ord);
            if (check.Count != 0)
            {
                string str = "";
                foreach (var i in check)
                    str += i.Name + ", ";
                throw new ArgumentException("Недостаточно следующих продуктов на складе:
" + str);
            }

            List<Product> allProds = GetProducts(ord);
            List<Product> allProdsT = GetStorageTickets(allProds);
            for (int i = 0; i < allProdsT.Count; i++ )
            {
                allProdsT[i].Spend(allProds[i].CountOnStorage);
            }
        }
```

```
        private List<Product> GetStorageTickets(List<Product> ls)
        {
            List<Product> res = new List<Product>();
            foreach (var i in ls)
            {
                Product ai = Content.Find(p => p.Name == i.Name);
                if (ai == null) { throw new ArgumentException("Не хватает товара на
складе "+i.Name); }
                res.Add(ai);
            }
            return res;
        }
        public double AppendAllQeue(int count)
        {
            double cost = 0;
            foreach (var i in ProductQuery)
            {
                    cost += i.Cost * count;
                    Append(i, count);
            }
            ClearQueue();
            return cost;
        }
        public double AppendAllQeue()
        {
            double cost = 0;
            foreach (var i in ProductQuery)
            {
                cost += i.Cost * i.CountOnStorage;
                Append(i, i.CountOnStorage);
            }
            ClearQueue();
            return cost;
        }
        public double AppendAllContent(int cnt)
        {
            double cost = 0;
            foreach (var i in Content)
            {
                Append(i, cnt);
                cost += i.Cost * cnt;
            }
            return cost;
        }
        public double AppendPosition(Product product, int cnt)
        {
            double cost = 0;
            cost = product.Cost * cnt;
            var selected = from p in Content
                           where p.Name == product.Name
                           select p;
            if (selected.Count() == 0)
            {
                product.SetCount(cnt);
                Content.Add(product);
            }
            else
            {
                selected.First().SetCount(selected.First().CountOnStorage + cnt);
            }
            return cost;
        }
        public double AppendAll(int count)
```

```
        {
            double cost = 0;
            foreach (var i in Content)
            {
                i.Append(count);
                cost += i.Cost * count;
            }
            return cost;
        }
        public double AppendPosition(string product, int cnt)
        {
            double cost = 0;

            var selected = from p in Content
                           where p.Name == product
                           select p;

            if (selected.Count() == 0)
            {
                throw new ArgumentException("Продукт не заведен на склад");
            }
            else
            {
                selected.First().SetCount(selected.First().CountOnStorage + cnt);
            }
            return cost;
        }
        private void ClearQueue()
        {
            List<Product> res = new List<Product>();
            foreach (var i in ProductQuery)
            {
                if (i.CountOnStorage > 0)
                    res.Add(i);
            }
            productQuery = res;
        }
    }
}
```