Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 ПРИКЛАДНАЯ ИНФОРМАТИКА**

О Т Ч Е Т

по лабораторной работе № 9

Название: Сериализация

Дисциплина: Разработка приложений на языке C#

Студент ИУ6-75Б _____ И.Ю. Жосан
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ А.М. Минитаева
(Подпись, дата) (И.О. Фамилия)

Москва, 2023

Задание:

Требуется разработать программу, ведущую учёт заказов в магазине. Классы:

— покупатель, с атрибутами: имя (string), адрес (string), скидка (double);

— товар: название (string) и цена (decimal);

— база данных товаров, хранящий ассоциативный массив («словарь») с информацией о товарах;

— orderLine с полями количество (int) и продукт (Product);

— order с полями номер заказа (int), клиент (Customer), скидка (decimal), общая стоимость (decimal) и строки заказа (List<OrderLIne>).

Реализовать следующую логику основной программы:

1. Создаётся и заполняется база данных товаров (ассоциативный массив).

2. В консоли вводятся данные по конкретному покупателю, создаётся соответствующий объект.

3. Создаётся заказ для введённого ранее покупателя. Устанавливается скидка на заказ в соответствии со скидкой покупателя.

4. В цикле формируются необходимое количество строк заказа: вводятся коды товаров и количества их единиц.

5. Полная информация о заказе сохраняется в файле с заданным именем.

Создать методы, которые осуществляют сериализацию/десериализацию объекта типа База данных товаров. Формат выбрать самостоятельно.

Выполнение задания:

Был разработан класс Клиент:

```
class Client
    {
        protected string name, addres;
        protected int diskont;
        public Client(string n, string a)
        {
            name = n;
            addres = a;
            Random r = new Random();
```

```
            diskont = r.Next(1,20);
        }
        public int Discont
        {
            get { return diskont; }
        }
        public string Name
        {
            get { return name; }
        }
        public string Addres
        {
            get { return addres; }
        }
    }
```

Класс Товар:

```
class Product
    {
        private string name;
        private decimal cost;
        public Product(string n, decimal c)
        {
            name = n;
            cost = c;
        }
        public string Name
        {
            get { return name; }
            set { name = value; }
        }
        public decimal Cost
        {
            get { return cost; }
            set { cost = value; }
        }
    }
```

Класс База товаров:

```
class ProductBase
   {
       static int globalNum = 1;
       protected Dictionary<int, Product> dict = new
                                  Dictionary<int, Product>();
       public ProductBase() { }
       public Dictionary<int, Product> Base
       {
           get { return dict; }
           set { dict = value; }
       }
       public void addProduct(Product p)
       {
           dict.Add(globalNum, p);
           globalNum++;
       }
       public Product getProduct(int code)
       {
           return dict[code];
       }
       public int maxCode()
       {
           return globalNum - 1;
       }
       public string ToString()
       {
           string result = "";
           for (int i = 0; i<dict.Count;i++)
               result += $"{i+1} -
                         {dict[i+1].Name}:\t{dict[i+1].Cost}\n";
           return result;
       }
   }
```

Класс Строка заказа:

```
class OrderLine
    {
        private int num;
        private Product prod;
        public OrderLine(int n, Product p)
        {
            num = n;
            prod = p;
        }
        public int Num
        {
            get { return num; }
        }
        public Product Prod
        {
            get { return prod; }
        }
}
```

Класс Заказ:

```
class Order
    {
        static int globalNum = 1;
        protected int num;
        protected Client client;
        protected double diskont;
        protected decimal all_cost = 0;
        protected List<OrderLine> order_lines = new
                                        List<OrderLine>();
        public Order(Client c)
        {
            num = globalNum;
            globalNum++;
            client = c;
            diskont = c.Discont;
        }
```

```
        public void addOrderLine(OrderLine l)
        {
            order_lines.Add(l);
            all_cost += l.Num * l.Prod.Cost * (100 -
                                    client.Discont ) / 100;
        }
        public int Num
        {
            get { return num; }
        }
        public Client Customer
        {
            get { return client; }
        }
        public double Diskont
        {
            get { return diskont; }
        }
        public List<OrderLine> OrderList
        {
            get { return order_lines; }
        }
        public decimal allCost
        {
            get { return all_cost; }
        }
    }
```

Основная программа:

```
static void Main(string[] args)
        {
            ProductBase Base = new ProductBase();
            Base.addProduct(new Product("Гречка", 56.5m));
            Base.addProduct(new Product("Молоко", 71.99m));
            Base.addProduct(new Product("Макароны", 40.0m));
            Base.addProduct(new Product("Хлеб", 39.99m));
```

```
Console.Write("Создание клиента:\nВведите имя:\t");
string name = Console.ReadLine();
Console.Write("Введите адрес:\t");
string address = Console.ReadLine();
Client cl = new Client(name, address);
Console.WriteLine($"Клиент создан. Ему присвоена скадка:
                                    {cl.Discont} %.");

Order or = new Order(cl);
Console.WriteLine("\nДобавьте в заказ продукт или
                                    завершье заказ:");
Console.WriteLine($"0 - Закончить\n{Base.ToString()}");
Console.Write("Введите код:\t");
int code = 0;
int.TryParse(Console.ReadLine(), out code);
while ((code != 0) && (code <= Base.maxCode()))
{
    Console.Write("Введите количество:\t");
    int num = 1;
    int.TryParse(Console.ReadLine(), out num);

    or.addOrderLine(new OrderLine(num,
                            Base.getProduct(code)));
    Console.WriteLine("\nДобавьте в заказ продукт или
                                    завершите заказ:");
    Console.WriteLine($"0 -
                        Закончить\n{Base.ToString()}");
    Console.Write("Введите код:\t");
    code = 0;
    int.TryParse(Console.ReadLine(), out code);
}

string json = JsonConvert.SerializeObject(or,
                                    Formatting.Indented);
```

```
        File.WriteAllText(@"C:\Users\Xiaomi\Desktop\C#\lab9_Serial
        ization\myJson.json", json);


          Console.WriteLine();
          json = JsonConvert.SerializeObject(Base);
          ProductBase Base2 =
             JsonConvert.DeserializeObject<ProductBase>(json);
          string json2 = JsonConvert.SerializeObject(Base2,
                                              Formatting.Indented);
          Console.WriteLine(json2);

          Console.ReadLine();
    }
```

Пример сохранения заказа в файле json на рисунке 1.

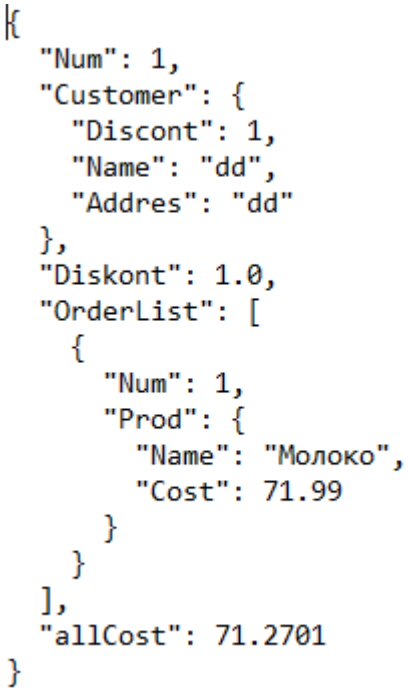

Рисунок 1 – Сохранение заказ в файле json

Проверка результата десериализации Базы товаров на рисунке 2.

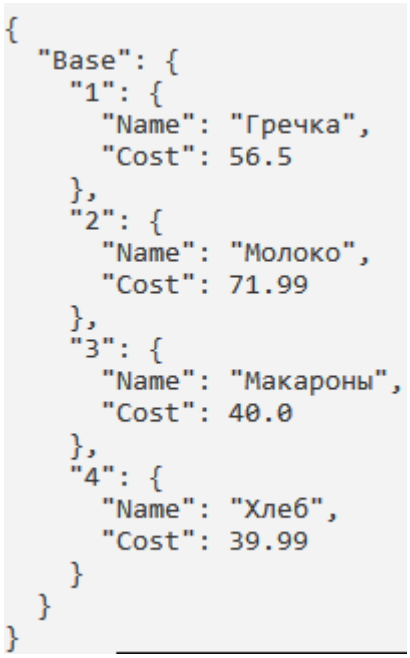

Рисунок 2 – Результат десериализации Базы товаров

Вывод: в процессе выполнения лабораторной работы были изучены средства сериализации и десериализации классов в C#.