МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННО БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Учебный Центр Информационных Технологий «Информатика»

Лабораторная работа № 2

по дисциплине «Визуальное программирование»

Направление подготовки: 230105 - «Программное обеспечение вычислительной техники и автоматизированных систем»

Выполнил слушатель: Бройтман Е.Д.
Вариант: №2
Дата сдачи:
Преподаватель: Силов Я.В.

Новосибирск, 2017г.

1. Задание

1.1. Создать структуру данных ИЗДЕЛИЕ: название, шифр, количество, комплектация.
1.2. Создать класс-список комплектации изделия деталями.
1.3. Создать класс детали.
1.4. Создать перечисление для типов деталей.
1.5. Перегрузить ToString() для класса-списка и класса детали.
1.6. Реализовать методы Add() и Remove() для класса-списка.
1.7. Для каждого поля реализовать методы Get() и Set().
1.8. Все поля данных должны иметь модификатор доступа private.
1.9. Для тестового класса создать отдельный файл с пользовательским меню.
1.10. Внутри классов «модели» не должно быть ввода/вывода на экран.

2. Структурное описание

В данной лабораторной работе создаются классы **Program, Product, Detail, CDetailNames, ListDetaillist** в пространстве имен lab2.
Для объявления и реализации методов и полей классов, в т.ч. методов Main использованы файлы «Program.cs», «Product.cs», «Detail.cs», «DetailNames.cs» и «ListDetaillist.cs».

Методы класса **Program**:
- private void ShowMenu() – метод вывода на экран главного меню программы.
- private void ShowMenuDet(int N) – подменю выбора типа детали.
- private void ShowMenuProd(Product Prod) – подменю выбора изменяемого параметра изделия.
- private void ShowMenuChangeDet() – подменю выбора способа изменения комплектации.
- private void ShowMenuChangeDetParams(int N, Product Prod) – подменю выбора изменяемых параметров деталей.
- private Product CreateProduct() – метод создания изделия.
- private void ChangeProduct(Product Prod) – метод внесения изменений в существующее изделие.
- static void Main(string[] args) – главный метод обработки действий пользователя.

Методы класса **Product**:
- public Product() – конструктор по-умолчанию.
- public Product(StringBuilder Name, StringBuilder Mark, int Quantity, ListDetaillist DetailList) – конструктор с параметрами.
- public StringBuilder GetName() – возвращает название (тип) изделия.
- public StringBuilder GetMark() – возвращает шифр (марку) изделия.
- public int GetQuantity() – возвращает количество изделий.
- public ListDetaillist GetDetailList() – возвращвет список деталей, комплектующих изделие.
- public void SetName(StringBuilder Name) – устанавливает название (тип) изделия.
- public void SetMark(StringBuilder Mark) – устанавливает шифр (марку) изделия.
- public void SetQuantity(int Quantity) – устанавливает количество изделий.

- public void SetDetailList(ListDetaillist DetailList) – устанавливает список деталей, комплектующих изделие.

Методы класса **Detail**:
- public Detail() – конструктор по-умолчанию.
- public Detail(CDetailNames.DetailName DName, StringBuilder DMark, double WeightInKg) – конструктор с параметрами.
- public Detail(Detail Det) – конструктор копирования.
- public CDetailNames.DetailName GetDName() – возвращает наименование (тип) детали.
- public StringBuilder GetDMark() – возвращает марку детали.
- public double GetWeight() – возвращвет вес детали.
- public void SetDName(CDetailNames.DetailName DName) – устанавливает наименование (тип) детали.
- public void SetDMark(StringBuilder DMark) – устанавливает марку детали.
- public void SetWeight(double WeightInKg) – устанавливает вес детали.
- public override string ToString() – перегрузка метода преобразования в строку.
- public bool Equals(Detail Det) – перегрузка метода проверки равенства объектов.

Класс **CDetailNames** служит контейнером для перечисления **DetailName**, и строкового static массива типов деталей DetailNames. Методов не имеет.

Методы класса **ListDetaillist**:
- public ListDetaillist() – конструктор по-умолчанию.
- public ListDetaillist(ListDetaillist DetailList) – конструктор копирования.
- public override string ToString() – перегрузка метода преобразования в строку.
- public new ListDetaillist Add(Detail Det) – метод добавления элемента в список.
- public new ListDetaillist Remove(Detail Det) – метод удаления элемента из списка.

3. Функциональное описание

Рассмотрим реализацию методов класса **Program**.
- private void ShowMenu() – построчный вывод пунктов меню и приглашения выбора пункта.
- private void ShowMenuDet(int N) – -//-.
- private void ShowMenuProd(Product Prod) – -//-.
- private void ShowMenuChangeDet() – -//-.
- private void ShowMenuChangeDetParams(int N, Product Prod) – -//-.
- private Product CreateProduct() – построчный ввод с консоли параметров изделия и его комплектующих.
- private void ChangeProduct(Product Prod) – редактирование параметров изделия по выбору.
- static void Main(string[] args) – в зависимости от выбора, сделанного в главном меню – добавление, изменение или просмотр параметров изделий и их комплектующих.

Рассмотрим реализацию методов класса **Product**:
- public Product() – пустой конструктор по-умолчанию.

- public Product(StringBuilder Name, StringBuilder Mark, int Quantity, ListDetaillist DetailList) – конструктор с параметрами присваивает соответствующие значения членам класса.
- public StringBuilder GetName() – возвращает название (тип) изделия.
- public StringBuilder GetMark() – возвращает шифр (марку) изделия.
- public int GetQuantity() – возвращает количество изделий.
- public ListDetaillist GetDetailList() – возвращвет список деталей, комплектующих изделие.
- public void SetName(StringBuilder Name) – устанавливает название (тип) изделия.
- public void SetMark(StringBuilder Mark) – устанавливает шифр (марку) изделия.
- public void SetQuantity(int Quantity) – устанавливает количество изделий.
- public void SetDetailList(ListDetaillist DetailList) – устанавливает список деталей, комплектующих изделие.

Рассмотрим реализацию методов класса **Detail**:
- public Detail() – пустой конструктор по-умолчанию.
- public Detail(CDetailNames.DetailName DName, StringBuilder DMark, double WeightInKg) – конструктор с параметрами присваивает соответствующие значения членам класса.
- public Detail(Detail Det) – конструктор копирования копирует значения членов входного объекта в вызывающий объект.
- public CDetailNames.DetailName GetDName() – возвращает наименование (тип) детали.
- public StringBuilder GetDMark() – возвращает марку детали.
- public double GetWeight() – возвращвет вес детали.
- public void SetDName(CDetailNames.DetailName DName) – устанавливает наименование (тип) детали.
- public void SetDMark(StringBuilder DMark) – устанавливает марку детали.
- public void SetWeight(double WeightInKg) – устанавливает вес детали.
- public override string ToString() – конкатенация полей класса через табуляцию.

public bool Equals(Detail Det) – проверка на равенство полей входного объекта полям вызывающего объекта.

Рассмотрим реализацию методов класса **ListDetaillist** (наследует встроенный класс **List<T>**):
- public ListDetaillist() – пустой конструктор по-умолчанию.
- public ListDetaillist(ListDetaillist DetailList) – поэлементно копирует в цикле поля входного класса.
- public override string ToString() – построчная конкатенация строк объектов-элементов списка.
- public new ListDetaillist Add(Detail Det) – добавление элементов в список с помощью метода Add базового класса.
- public new ListDetaillist Remove(Detail Det) – удаление элементов списка с помощью метода Remove базового класса.

4. Описание работы программы

После запуска программы на экране появляется главное меню.

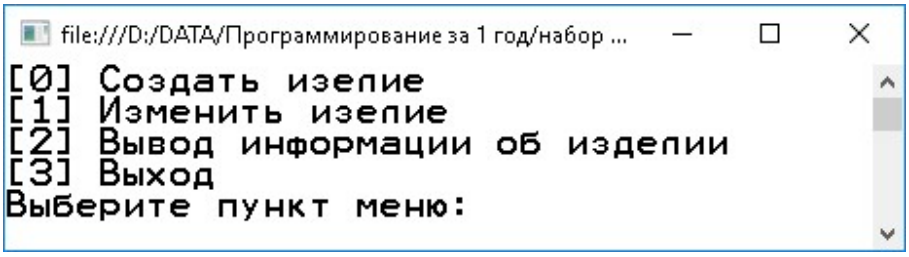

Рис. 1 Главное меню.

После ввода п. 0, программа предлагает ввести значения для создаваемого изделия.

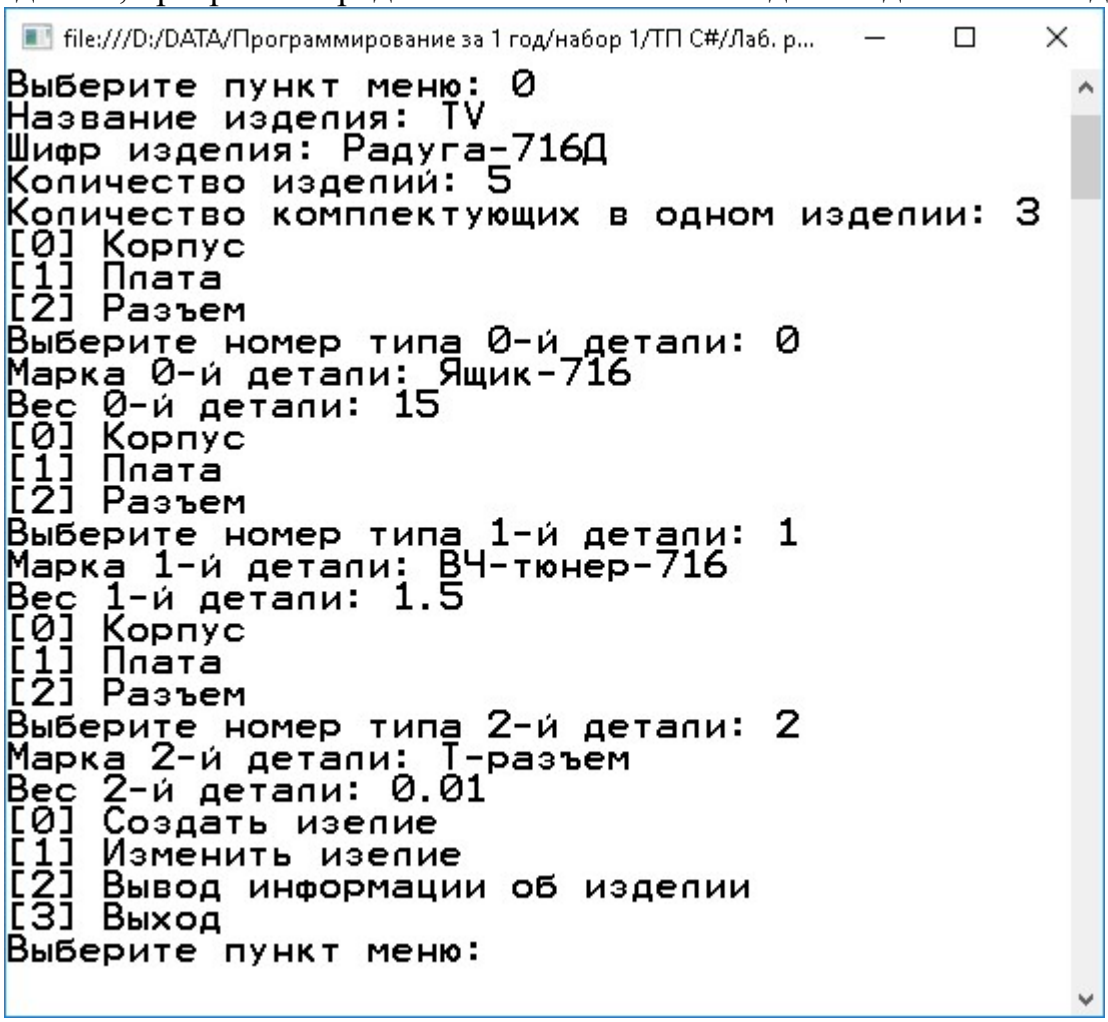

Рис. 2 Создание изделия.

После выбора п. 1, программа предлагает внести изменения в изделие.

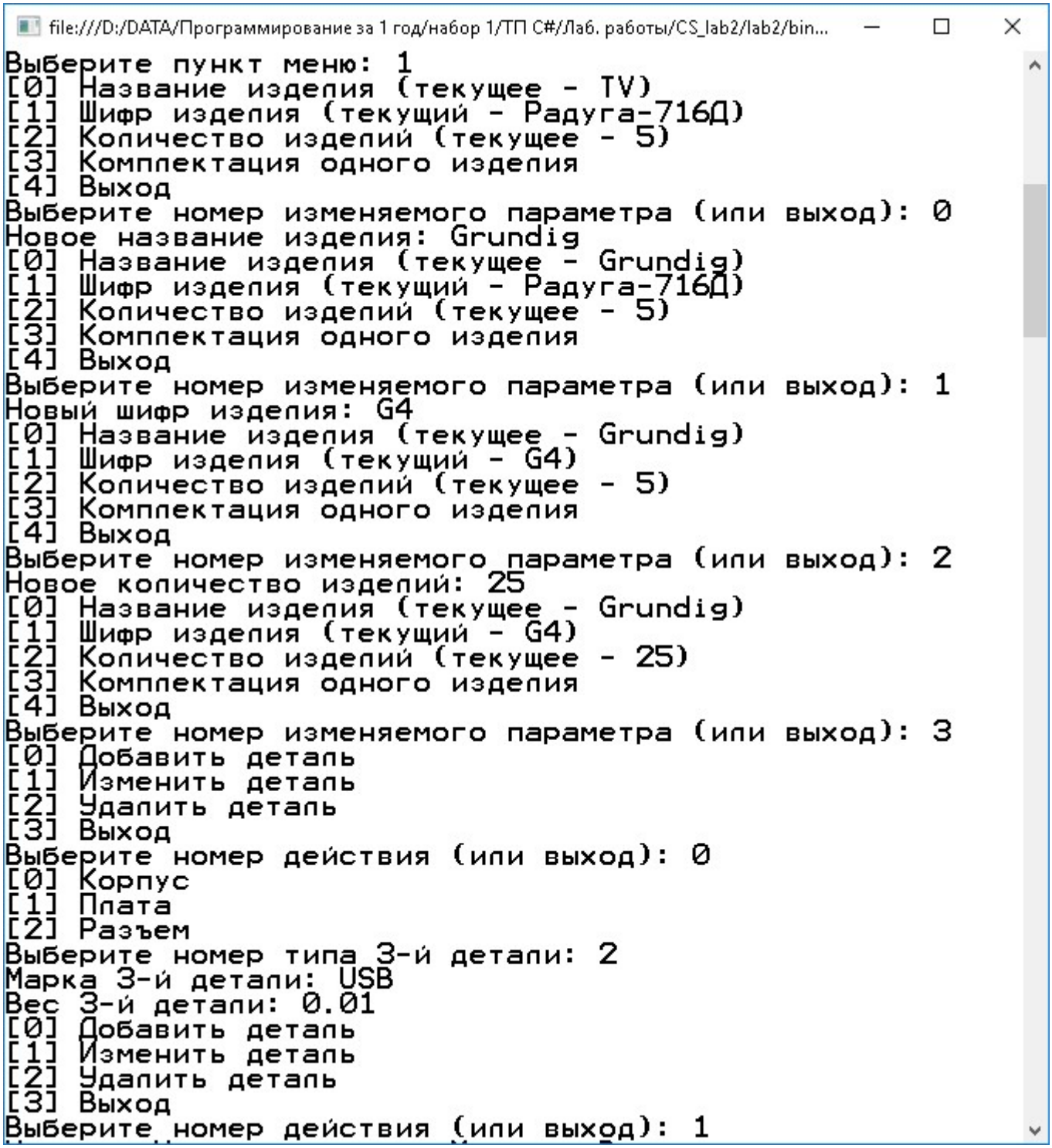

Рис. 3 Изменение изделия (ч. 1).

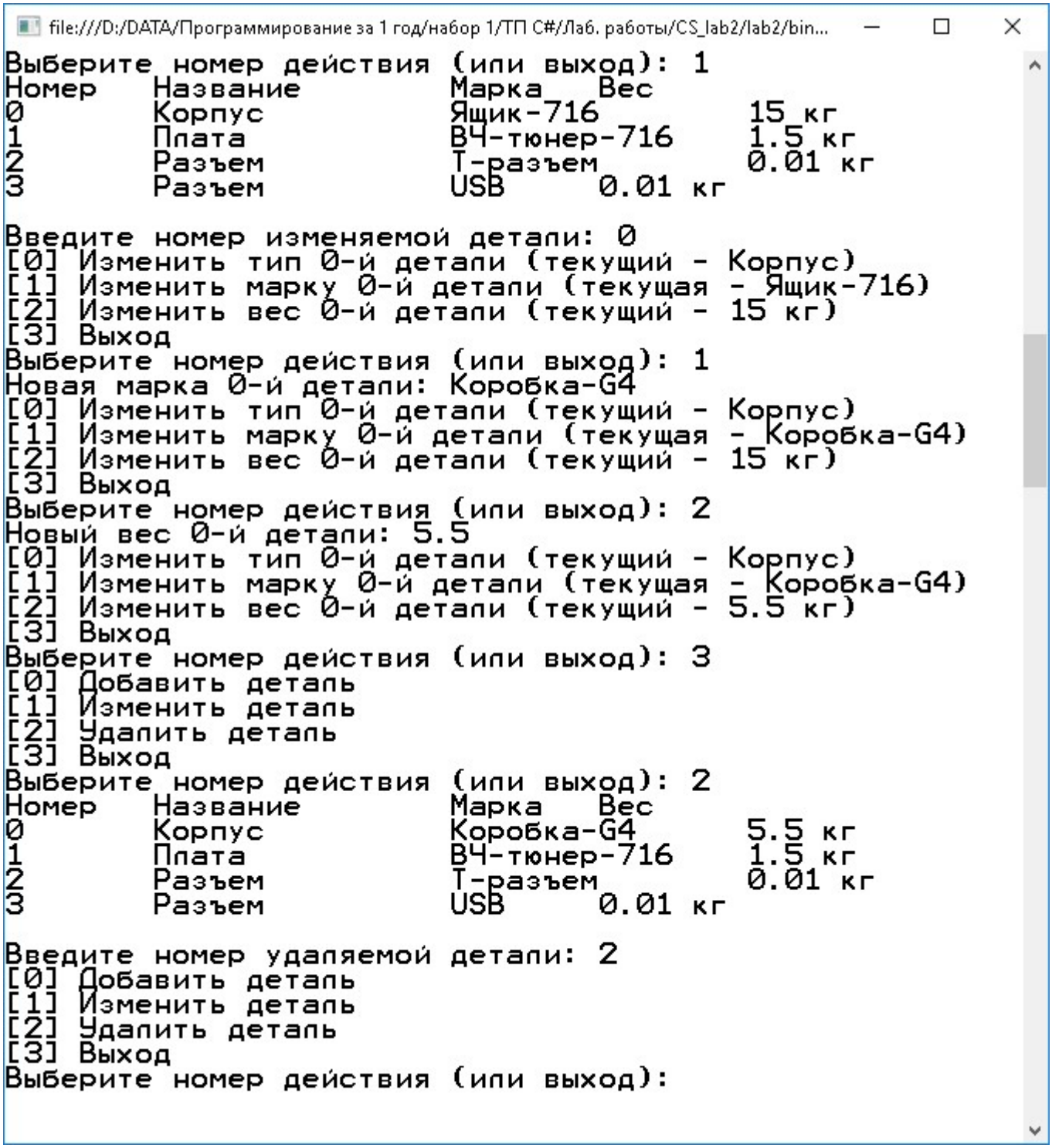

Рис. 4 Изменение изделия (ч. 2).

После выбора п. 2 главного меню, программа выводит информацию об изделии.

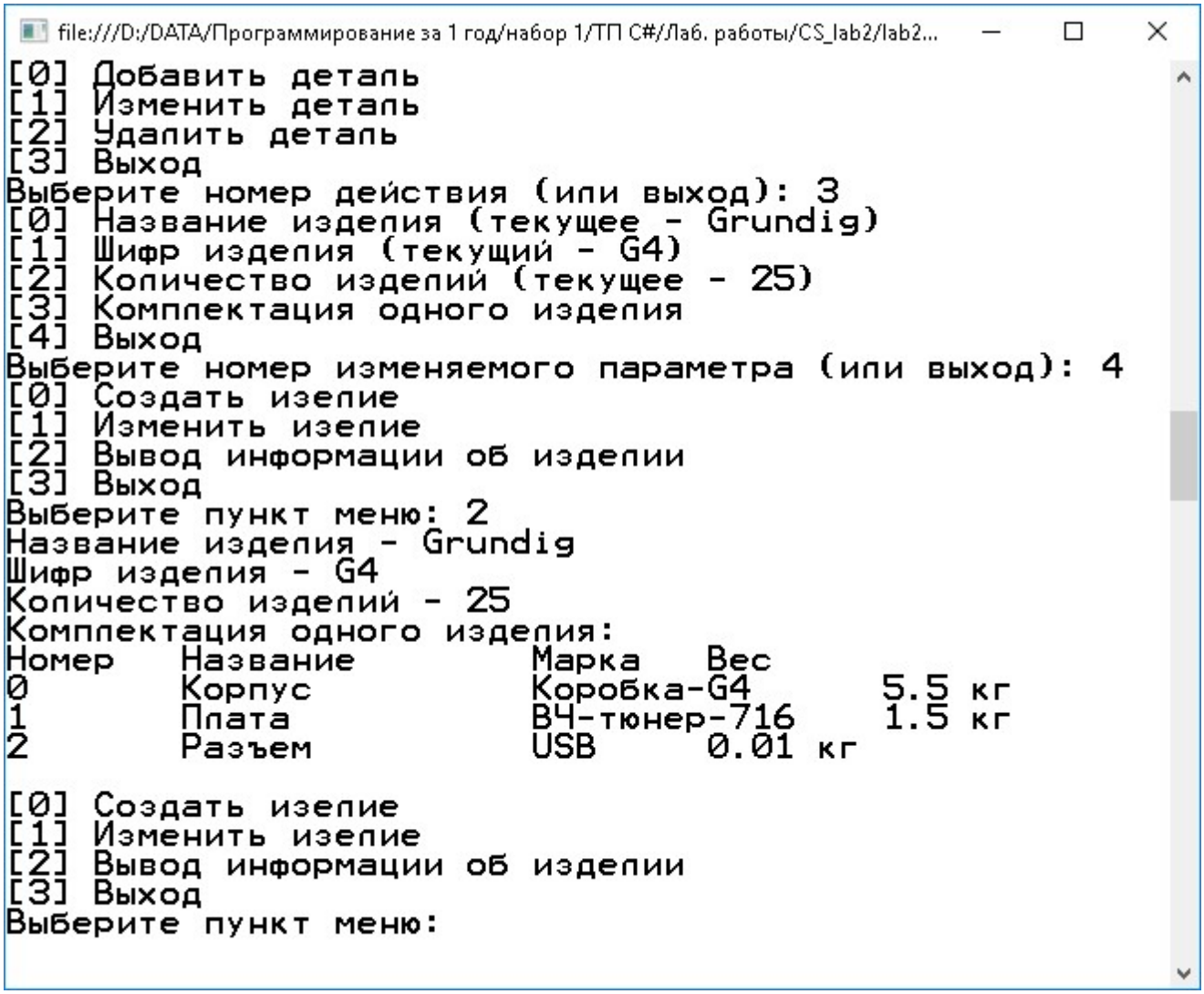

Рис. 5 Информация об изделии.

После выбора п. 3 главного меню, происходит выход из программы.

5. Выводы

В результате выполнения данной лабораторной работы, получен практический опыт работы с классами языка C#. Целью данной работы является:

- познакомится с пользовательскими типами данных в языке C#: структура и перечисление.
- ознакомиться со структурой класса, его созданием и использованием, описанием членов класса: полей, свойств, инициализации объектов класса с помощью конструкторов.

В данной лабораторной работе создано пространство имен lab2 с классами **Program, Product, Detail, CDetailNames, ListDetaillist**. Для каждого класса, кроме **CDetailNames,** определены соответствующие методы:

- метод вывода на экран главного меню программы.
- подменю выбора типа детали.
- подменю выбора изменяемого параметра изделия.
- подменю выбора способа изменения комплектации.
- подменю выбора изменяемых параметров деталей.
- метод создания изделия.
- метод внесения изменений в существующее изделие.
- главный метод обработки действий пользователя.

- конструкторы по-умолчанию.
- конструкторы с параметрами.
- конструкторы копирования.
- метод определения названия (типа) изделия.
- метод определения шифра (марки) изделия.
- метод определения количества изделий.
- метод определения списка деталей, комплектующих изделие.
- метод установки названия (типа) изделия.
- метод установки шифра (марки) изделия.
- метод установки количества изделий.
- метод установки списка деталей, комплектующих изделие.
- метод определения наименования (тип) детали.
- метод определения марки детали.
- метод определения веса детали.
- метод установки наименования (типа) детали.
- метод установки марки детали.
- метод установки веса детали.
- перегрузка методов преобразования в строку.
- перегрузка метода проверки равенства объектов.
- метод добавления элемента в список.
- метод удаления элемента из списка.

Файлы программы находятся в репозитории по адресу: https://github.com/broitman-eugeny/CS_lab2.

6. Код программы с комментариями

“Program.cs ”

```
//Визуальное программирование. Лабораторная работа №2
//Вариант 2
//1. Для заданной структуры данных разработать абстрактный класс и класс наследник.
//В классе реализовать несколько конструкторов. Создать методы, работающие с полями класса.
//Часть из них должны быть виртуальными. Добавить методы-свойства и индексаторы.
//2. Разработать интерфейсные классы, добавляющие некоторые методы в класс-потомок.
//Изучить причины возникновения коллизий имен при наследовании и способы устранения.
//3. Разработать классы исключительных ситуаций и применить их для обработки, возникающих исключе-
ний.
//4. Написать демонстрационную программу.

// Структура данных:
//ИЗДЕЛИЕ: название, шифр, количество, комплектация.
using System;
using System.Text;

namespace lab2
{
    class Program
    {
        private void ShowMenu()
        {
            Console.WriteLine("[0] Создать изелие");
            Console.WriteLine("[1] Изменить изелие");
            Console.WriteLine("[2] Вывод информации об изделии");
            Console.WriteLine("[3] Выход");
            Console.Write("Выберите пункт меню: ");
        }
        private void ShowMenuDet(int N)
        {
            for (int i = 0; i < CDetailNames.count; i++)
            {
                Console.WriteLine("[{0}] {1}", i, CDetailNames.DetailNames[i]);
            }
            Console.Write("Выберите номер типа {0}-й детали: ", N);
        }
        private void ShowMenuProd(Product Prod)
        {
            Console.WriteLine("[0] Название изделия (текущее - {0})", Prod.GetName());
            Console.WriteLine("[1] Шифр изделия (текущий - {0})", Prod.GetMark());
            Console.WriteLine("[2] Количество изделий (текущее - {0})", Prod.GetQuantity());
            Console.WriteLine("[3] Комплектация одного изделия");
            Console.WriteLine("[4] Выход");
            Console.Write("Выберите номер изменяемого параметра (или выход): ");
        }
        private void ShowMenuChangeDet()
        {
            Console.WriteLine("[0] Добавить деталь");
            Console.WriteLine("[1] Изменить деталь");
            Console.WriteLine("[2] Удалить деталь");
            Console.WriteLine("[3] Выход");
            Console.Write("Выберите номер действия (или выход): ");
        }
        private void ShowMenuChangeDetParams(int N, Product Prod)
        {
            Console.WriteLine("[0] Изменить тип {0}-й детали (текущий - {1})", N,
CDetailNames.DetailNames[(int)(Prod.GetDetailList()[N].GetDName())]);
            Console.WriteLine("[1] Изменить марку {0}-й детали (текущая - {1})", N,
Prod.GetDetailList()[N].GetDMark());
            Console.WriteLine("[2] Изменить вес {0}-й детали (текущий - {1} кг)", N,
Prod.GetDetailList()[N].GetWeight());
            Console.WriteLine("[3] Выход");
            Console.Write("Выберите номер действия (или выход): ");
        }
        //Создать изделие
        private Product CreateProduct()
        {
```

```
            Console.Write("Название изделия: ");
            StringBuilder Name = new StringBuilder(Console.ReadLine());
            Console.Write("Шифр изделия: ");
            StringBuilder Mark = new StringBuilder(Console.ReadLine());
            Console.Write("Количество изделий: ");
            string SQuantity = Console.ReadLine();
            int Quantity;
            while (int.TryParse(SQuantity, out Quantity) != true)
            {
                Console.Write("Неверный ввод. Еще раз: ");
                SQuantity = Console.ReadLine();
            }
            Console.Write("Количество комплектующих в одном изделии: ");
            string SDQuantity = Console.ReadLine();
            int DQuantity;
            while (int.TryParse(SDQuantity, out DQuantity) != true)
            {
                Console.Write("Неверный ввод. Еще раз: ");
                SDQuantity = Console.ReadLine();
            }
            ListDetaillist DetailList = new ListDetaillist();
            for (int i = 0; i < DQuantity; i++ )
            {
                //Выбор типа детали
                ShowMenuDet(i);
                string SNDet = Console.ReadLine();
                int NDet;
                while (int.TryParse(SNDet, out NDet) != true ||
                       NDet<0 || NDet>= CDetailNames.count)
                {
                    Console.Write("Неверный ввод. Еще раз: ");
                    SNDet = Console.ReadLine();
                }
                //Ввод марки детали
                Console.Write("Марка {0}-й детали: ", i);
                StringBuilder DMark = new StringBuilder(Console.ReadLine());
                //Ввод веса детали
                Console.Write("Вес {0}-й детали: ", i);
                string SWeightInKg = Console.ReadLine();
                double WeightInKg;
                while (double.TryParse(SWeightInKg, out WeightInKg) != true)
                {
                    Console.Write("Неверный ввод. Еще раз: ");
                    SWeightInKg = Console.ReadLine();
                }
                //Создание детали
                Detail Det = new Detail((CDetailNames.DetailName)NDet, DMark, WeightInKg);
                DetailList.Add(Det);
            }
            Product Prod = new Product(Name, Mark, Quantity, DetailList);
            return Prod;
        }//private Product CreateProduct()
        //Изменить изделие
private void ChangeProduct(Product Prod)
        {
            int C = -1;
            while (C != '4')
            {
                ShowMenuProd(Prod);
                C = Console.Read();
                Console.ReadLine();
                switch (C)
                {
                    case '0'://Название изделия
                        Console.Write("Новое название изделия: ");
                        StringBuilder Name = new StringBuilder(Console.ReadLine());
                        Prod.SetName(Name);
                        break;
                    case '1'://Шифр изделия
```

```
                    Console.Write("Новый шифр изделия: ");
                    StringBuilder Mark = new StringBuilder(Console.ReadLine());
                    Prod.SetMark(Mark);
                    break;
                case '2'://Количество изделий
                    Console.Write("Новое количество изделий: ");
                    string SQuantity = Console.ReadLine();
                    int Quantity;
                    while (int.TryParse(SQuantity, out Quantity) != true)
                    {
                        Console.Write("Неверный ввод. Еще раз: ");
                        SQuantity = Console.ReadLine();
                    }
                    Prod.SetQuantity(Quantity);
                    break;
                case '3'://Комплектация
                    int C1 = -1;
                    while (C1 != '3')
                    {
                        ShowMenuChangeDet();
                        C1 = Console.Read();
                        Console.ReadLine();
                        int DIndex;
                        switch (C1)
                        {
                            case '0'://Добавить деталь
                                //Выбор типа детали
                                DIndex = Prod.GetDetailList().Count;//Индекс добавляемой детали
                                ShowMenuDet(DIndex);
                                string SNDet = Console.ReadLine();
                                int NDet;
                                while (int.TryParse(SNDet, out NDet) != true)
                                {
                                    Console.Write("Неверный ввод. Еще раз: ");
                                    SNDet = Console.ReadLine();
                                }
                                //Ввод марки детали
                                Console.Write("Марка {0}-й детали: ", DIndex);
                                StringBuilder DMark = new StringBuilder(Console.ReadLine());
                                //Ввод веса детали
                                Console.Write("Вес {0}-й детали: ", DIndex);
                                string SWeightInKg = Console.ReadLine();
                                double WeightInKg;
                                while (double.TryParse(SWeightInKg, out WeightInKg) != true)
                                {
                                    Console.Write("Неверный ввод. Еще раз: ");
                                    SWeightInKg = Console.ReadLine();
                                }
                                //Создание детали
                                Detail Det = new Detail((CDetailNames.DetailName)NDet, DMark,
WeightInKg);
                                Prod.GetDetailList().Add(Det);
                                break;
                            case '1'://Изменить деталь
                                Console.WriteLine(Prod.GetDetailList().ToString());
                                Console.Write("Введите номер изменяемой детали: ");
                                string SDIndex = Console.ReadLine();
                                while (int.TryParse(SDIndex, out DIndex) != true ||
                                       DIndex<0 || DIndex>=Prod.GetDetailList().Count)
                                {
                                    Console.Write("Неверный ввод. Еще раз: ");
                                    SDIndex = Console.ReadLine();
                                }
                                int C2 = -1;
                                while (C2 != '3')
                                {
                                    ShowMenuChangeDetParams(DIndex, Prod);
                                    C2 = Console.Read();
                                    Console.ReadLine();
```

```
                                        switch (C2)
                                        {
                                            case '0'://Изменить тип
                                                ShowMenuDet(DIndex);
                                                SNDet = Console.ReadLine();
                                                while (int.TryParse(SNDet, out NDet) != true)
                                                {
                                                    Console.Write("Неверный ввод. Еще раз: ");
                                                    SNDet = Console.ReadLine();
                                                }

Prod.GetDetailList()[DIndex].SetDName((CDetailNames.DetailName)NDet);
                                                break;
                                            case '1'://Изменить марку
                                                Console.Write("Новая марка {0}-й детали: ", DIndex);
                                                DMark = new StringBuilder(Console.ReadLine());
                                                Prod.GetDetailList()[DIndex].SetDMark(DMark);
                                                break;
                                            case '2'://Изменить вес
                                                Console.Write("Новый вес {0}-й детали: ", DIndex);
                                                SWeightInKg = Console.ReadLine();
                                                while (double.TryParse(SWeightInKg, out WeightInKg)
!= true ||
                                                       WeightInKg<0)
                                                {
                                                    Console.Write("Неверный ввод. Еще раз: ");
                                                    SWeightInKg = Console.ReadLine();
                                                }
                                                Prod.GetDetailList()[DIndex].SetWeight(WeightInKg);
                                                break;
                                            case '3'://Выход
                                                break;
                                            default:
                                                break;
                                        }
                                    }//while (C2 != '3')
                                    break;
                                case '2'://Удалить деталь
                                    Console.WriteLine(Prod.GetDetailList().ToString());
                                    Console.Write("Введите номер удаляемой детали: ");
                                    SDIndex = Console.ReadLine();
                                    while (int.TryParse(SDIndex, out DIndex) != true ||
                                           DIndex < 0 || DIndex >= Prod.GetDetailList().Count)
                                    {
                                        Console.Write("Неверный ввод. Еще раз: ");
                                        SDIndex = Console.ReadLine();
                                    }
                                    Prod.GetDetailList().Remove(Prod.GetDetailList()[DIndex]);
                                    break;
                                case '3'://Выход
                                    break;
                                default:
                                    break;
                            }//switch (C1)
                        }//while (C1 != '3')
                        break;//case '3'://Комплектация
                    case '4'://Выход
                        break;
                    default:
                        break;
                }//switch (C)
            }//while (C != '4')
        }//private Product ChangeProduct(Product Prod)
static void Main(string[] args)
        {
            Program Prog = new Program();
            Product Prod=null;
            int C=-1;
            while (C != '3')
```

```
            {
                Prog.ShowMenu();
                C=Console.Read();
                Console.ReadLine();
                switch (C)
                {
                    case '0'://Создать изелие
                        Prod = Prog.CreateProduct();
                        break;
                    case '1'://Изменить изелие
                        if (Prod != null)
                        {
                            Prog.ChangeProduct(Prod);
                        }
                        else
                        {
                            Console.WriteLine("Сначала создайте изделие!");
                        }
                        break;
                    case '2'://Вывод информации об изделии
                        if (Prod != null)
                        {
                            Console.WriteLine("Название изделия - {0}", Prod.GetName());
                            Console.WriteLine("Шифр изделия - {0}", Prod.GetMark());
                            Console.WriteLine("Количество изделий - {0}", Prod.GetQuantity());
                            Console.WriteLine("Комплектация одного изделия:");
                            Console.WriteLine(Prod.GetDetailList().ToString());
                        }
                        else
                        {
                            Console.WriteLine("Сначала создайте изделие!");
                        }
                        break;
                    case '3'://Выход
                        break;
                    default:
                        break;
                }
            }
        }//static void Main(string[] args)
    }//class Program
}//namespace lab2
```

“Product.cs ”

```
//Класс продукт
using System.Text;

namespace lab2
{
    class Product //ИЗДЕЛИЕ
    {
        StringBuilder Name; //название
        StringBuilder Mark; //шифр
        int Quantity; //количество
        ListDetaillist DetailList; //комплектация

        public Product() { }
        public Product(StringBuilder Name, StringBuilder Mark, int Quantity, ListDetaillist
DetailList)
        {
            this.Name = new StringBuilder(Name.ToString());
            this.Mark = new StringBuilder(Mark.ToString());
            this.Quantity = Quantity;
            this.DetailList = new ListDetaillist(DetailList);
        }
        public StringBuilder GetName() { return Name; }
        public StringBuilder GetMark() { return Mark; }
        public int GetQuantity() { return Quantity; }
        public ListDetaillist GetDetailList() { return DetailList; }
        public void SetName(StringBuilder Name) { this.Name = new StringBuilder(Name.ToString()); }
```

```
        public void SetMark(StringBuilder Mark) { this.Mark = new StringBuilder(Mark.ToString()); }
        public void SetQuantity(int Quantity) { this.Quantity = Quantity; }
        public void SetDetailList(ListDetaillist DetailList) { this.DetailList = new
ListDetaillist(DetailList); }
    }
}
```

“ Detail.cs ”

```
//Класс Деталь
using System;
using System.Text;

namespace lab2
{
    class Detail : IEquatable<Detail>
    {
        CDetailNames.DetailName DName;
        StringBuilder DMark;
        double WeightInKg;

        public Detail() { }
        public Detail(CDetailNames.DetailName DName, StringBuilder DMark, double WeightInKg)
        {
            this.DName = DName;
            this.DMark = new StringBuilder(DMark.ToString());
            this.WeightInKg = WeightInKg;
        }
        public Detail(Detail Det)
        {
            this.DName = Det.DName;
            this.DMark = new StringBuilder(Det.DMark.ToString());
            this.WeightInKg = Det.WeightInKg;
        }
        public CDetailNames.DetailName GetDName() { return DName; }
        public StringBuilder GetDMark() { return DMark; }
        public double GetWeight() { return WeightInKg; }
        public void SetDName(CDetailNames.DetailName DName) { this.DName = DName; }
        public void SetDMark(StringBuilder DMark) { this.DMark = new
StringBuilder(DMark.ToString()); }
        public void SetWeight(double WeightInKg) { this.WeightInKg = WeightInKg; }
        public override string ToString()
        {
            return new string(
                              (CDetailNames.DetailNames[(int)DName]+
                                "\t\t"+
                                DMark.ToString()+
                                "\t"+
                                WeightInKg.ToString()+
                                " кг"
                               ).ToCharArray()
                              );
        }
        public bool Equals(Detail Det)
        {
            if (Det == null) return false;
            if (
                this.DName == Det.DName &&
                this.DMark.ToString().Equals(Det.DMark.ToString()) &&
                this.WeightInKg == Det.WeightInKg
               )
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }
}
```

“ DetailNames.cs ”

```
//Перечисление Название детали
namespace lab2
{
    class CDetailNames
    {
        public enum DetailName
        {
            Carcass,
            Board,
            Socket
        }
        public static int count = 3;
        public static string[] DetailNames = {"Корпус","Плата","Разъем" };
    }
}
```

“ListDetaillist.cs ”

```
//Класс список деталей
using System.Collections.Generic;
using System.Text;

namespace lab2
{
    class ListDetaillist:List<Detail>
    {
        public ListDetaillist() { }
        public ListDetaillist(ListDetaillist DetailList)
        {
            foreach (var CurDet in DetailList)
            {
                this.Add(CurDet);
            }
        }
        public override string ToString()
        {
            StringBuilder STmp = new StringBuilder("Номер\tНазвание\tМарка\tВес\n");
            for(int i=0; i< this.Count; i++)
            {
                STmp.Append(string.Format("{0}\t",i)+this[i].ToString()+"\n");
            }
            return STmp.ToString();
        }
        public new ListDetaillist Add(Detail Det)
        {
            Detail DTemp = new Detail(Det);
            base.Add(DTemp);
            return this;
        }
        public new ListDetaillist Remove(Detail Det)
        {
            base.Remove(Det);
            return this;
        }
    }
}
```