

НИУ ВШЭ

# Пояснительная записка к контрольному домашнему заданию

Трофимов Илья, 172ПИ. Вариант 44

2013 год

<b>1. УСЛОВИЕ ЗАДАЧИ .....</b>	<b>3</b>
<b>2    ФУНКЦИИ РАЗРАБАТЫВАЕМОГО ПРИЛОЖЕНИЯ .....</b>	<b>4</b>
<b>2.1. ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ.....</b>	<b>4</b>
<b>2.2. ОПИСАНИЕ ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ .....</b>	<b>4</b>
2.2.1. Меню «Файл».....	4
2.2.2. Меню «Редактировать» .....	5
2.2.3. Область отображения списка .....	5
2.2.4. Область редактирования элементов .....	6
<b>3.    СТРУКТУРА ПРИЛОЖЕНИЯ .....</b>	<b>7</b>
3.1. ДИАГРАММА КЛАССОВ.....	7
3.2. ОПИСАНИЕ КЛАССОВ, ИХ ПОЛЕЙ И МЕТОДОВ.....	7
<b>4.    РАСПРЕДЕЛЕНИЕ ИСХОДНОГО КОДА ПО ФАЙЛАМ ПРОЕКТА.....</b>	<b>10</b>
<b>5.    ТЕКСТ ПРОГРАММЫ .....</b>	<b>11</b>
<b>6.    СПИСОК ЛИТЕРАТУРЫ .....</b>	<b>23</b>

## 1. УСЛОВИЕ ЗАДАЧИ

### Вариант 44

Определить базовый и производные классы:

- Сладость – абстрактный класс (наименование, страна производитель, стоимость за кг, температура хранения, срок годности, калорийность)
- Мороженое (из чего изготовлено, дата производства, другие члены класса выбрать самостоятельно)
- Курага (дата производства, другие члены класса выбрать самостоятельно)

При организации взаимоотношений между классами в иерархии наследования использовать в базовом классе виртуальные методы, переопределяя и используя их в производных классах.

Общий вид записи об учетной записи в файле:

**==<Вид\_сладости>==**

**\*\***

**<Данные об объекте>**

## 2. ФУНКЦИИ РАЗРАБАТЫВАЕМОГО ПРИЛОЖЕНИЯ

### 2.1. ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ

Данное ПО может использоваться для ведения списка кондитерских изделий на продуктовых предприятиях, позволяя отслеживать текущее состояние склада.

### 2.1. ОПИСАНИЕ ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ

#### 2.1.1. Меню «Файл»

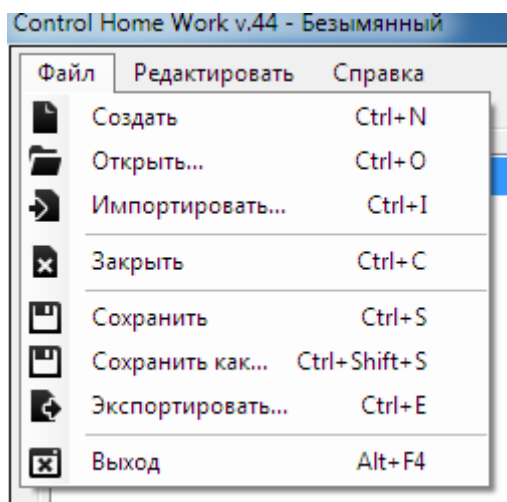


Рисунок 1 . Меню «Файл»

- **Создать** – создать новый список;
- **Открыть** – открыть список из текстового файла;
- **Импортировать** – добавить данные из уже существующего текстового файла к текущему списку;
- **Закреть** – закрыть текущий список;
- **Сохранить** – сохранить текущий список в текстовый файл;
- **Сохранить как** – сохранить текущий список в новый текстовый файл;
- **Экспортировать** – добавить данные из текущего списка к уже существующему текстовому файлу;
- **Выход** – завершить работу программы.

### 2.1.2. Меню «Редактировать»

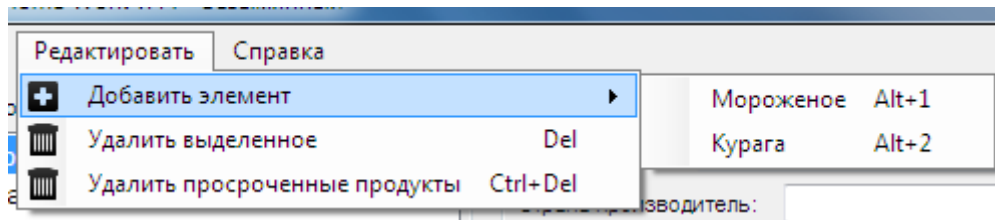


Рисунок 2. Меню «Редактировать»

- **Добавить элемент** – выпадающий список;
  - **Мороженое** – добавить мороженое в текущий список;
  - **Курага** – добавить курагу в текущий список;
- **Удалить выделенное** – удалить выделенные элементы из списка;
- **Удалить просроченные продукты** – удалить из списка просроченные продукты;

### 2.1.3. Область отображения списка

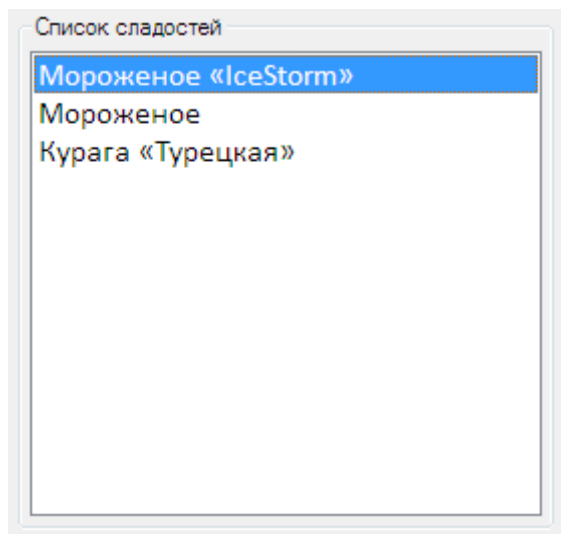
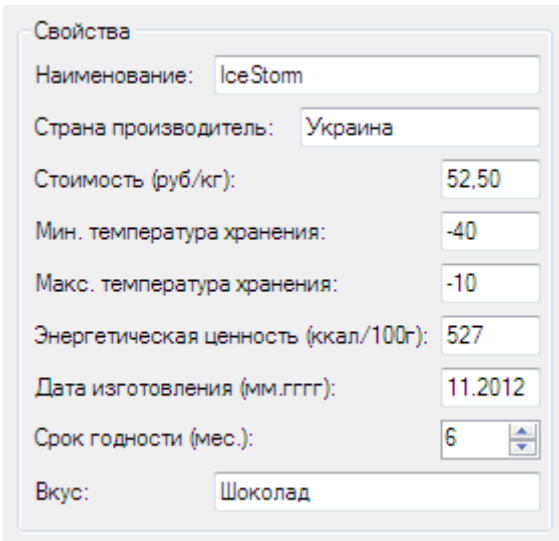


Рисунок 3. Область отображения списка

Реализована через ListBox.

#### 2.1.4. Область редактирования элементов



Свойства

Наименование:	IceStom
Страна производитель:	Украина
Стоимость (руб./кг):	52,50
Мин. температура хранения:	-40
Макс. температура хранения:	-10
Энергетическая ценность (ккал/100г):	527
Дата изготовления (мм.гггг):	11.2012
Срок годности (мес.):	6
Вкус:	Шоколад

Рисунок 4. Область редактирования элементов

Представлена через компоненты TextBox и NumericUpDown.

### 3. СТРУКТУРА ПРИЛОЖЕНИЯ

#### 3.1. ДИАГРАММА КЛАССОВ

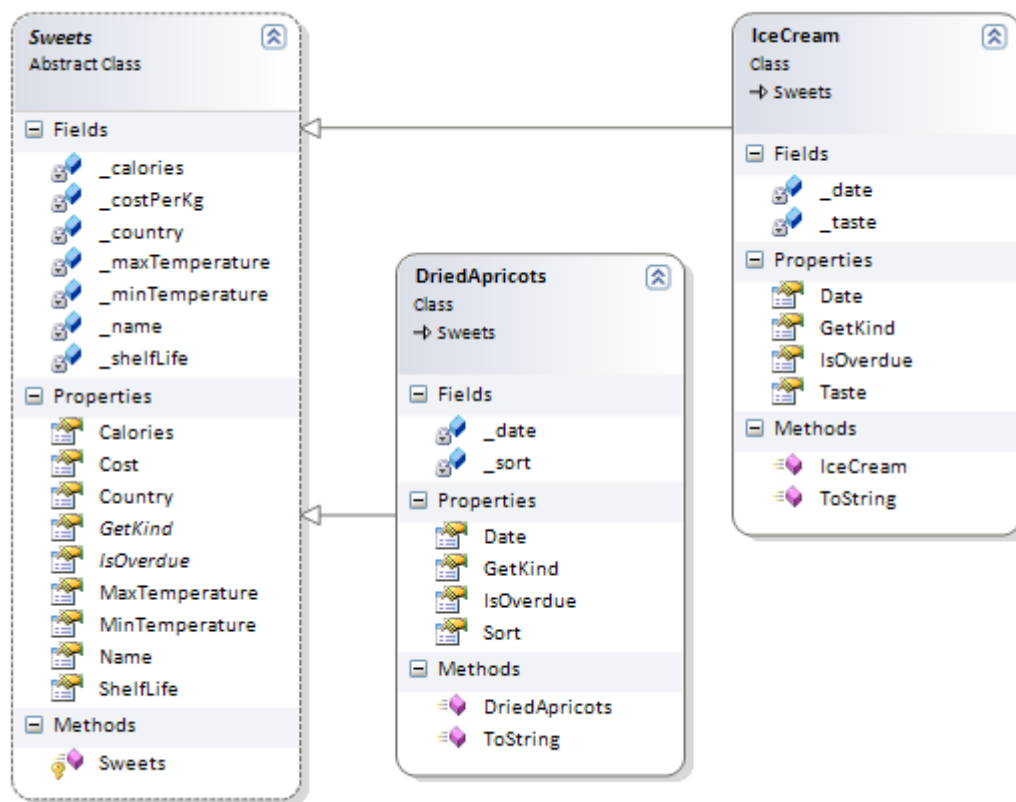


Рисунок 5. Диаграмма классов

#### 3.2. ОПИСАНИЕ КЛАССОВ, ИХ ПОЛЕЙ И МЕТОДОВ

- **Sweets** – базовый, абстрактный класс;
  - **Закрытые поля класса Sweets:**
    - **string \_name** – наименование;
    - **string \_country** – страна производитель;
    - **double \_costPerKg** – стоимость за килограмм;
    - **int \_minTemperature** – минимальная температура хранения;
    - **int \_maxTemperature** – максимальная температура хранения;
    - **int \_calories** – калорийность
    - **uint \_shelfLife** – срок годности;

- **Открытые свойства класса Sweets для доступа к закрытым полям класса:**
    - **string Name;**
    - **string Country;**
    - **double Cost;**
    - **int MinTemperature;**
    - **int MaxTemperature;**
    - **int Calories;**
    - **uint ShelfLife;**
  - **Абстрактные свойства класса Sweets:**
    - **bool IsOverdue;**
    - **string GetKind;**
  - **Конструктор по умолчанию.**
- 
- **IceCream – производный от Sweets класс;**
    - **Закрытые поля класса IceCream:**
      - **string \_date** – дата изготовления;
      - **string \_taste** – вкус;
    - **Открытые свойства класса IceCream для доступа к закрытым полям класса:**
      - **string Date;**
      - **string Taste;**
    - **Переопределенные свойства:**
      - **bool IsOverdue** – Возвращает true если продукт просрочен и false если нет или невозможно узнать.
      - **string GetKind;**
    - **Конструктор по умолчанию.**



- **DriedApricots** – производный от Sweets класс;
  - **Закрытые поля класса DriedApricots:**
    - **string \_date** – дата изготовления;
    - **string \_sort** – сорт кураги;
  - **Открытые свойства класса DriedApricots для доступа к закрытым полям класса:**
    - **string Date;**
    - **string Sort;**
  - **Переопределенные свойства:**
    - **bool IsOverdue** – Возвращает true если продукт просрочен и false если нет или невозможно узнать.
    - **string GetKind;**
  - **Конструктор по умолчанию.**

#### 4. РАСПРЕДЕЛЕНИЕ ИСХОДНОГО КОДА ПО ФАЙЛАМ ПРОЕКТА

- **SweetLibrary;**
  - **Sweets.cs** – содержит описание класса «Сладости»
  - **IceCream.cs** – содержит описание класса «Мороженое»
  - **DriedApricots.cs** – содержит описание класса «Курага»
- **SweetForms;**
  - **MainWindow.cs** – главное окно
  - **Methods.cs** – статический класс, содержащий ряд методов необходимых для главной формы
- **CommonWorkingLibrary;**
  - **Literals.cs** – статический класс, содержащий все строковые константы используемые в программе

## 5. ТЕКСТ ПРОГРАММЫ

### Sweets.cs

```
using System;
using CommonWorkingLibrary;

namespace SweetLibrary
{
    public abstract class Sweets // Базовый класс "Сладости"
    {
        private string _name; // Наименование
        private string _country; // Страна производитель
        private double _costPerKg; // Стоимость за килограмм
        private int _minTemperature; // Минимальная температура хранения
        private int _maxTemperature; // Максимальная температура хранения
        private int _calories; // Калорийность
        private uint _shelfLife; // Срок годности

        protected Sweets()
        {
            _name = String.Empty;
            _country = String.Empty;
            _costPerKg = 0;
            _minTemperature = -101;
            _maxTemperature = 101;
            _calories = -1;
            _shelfLife = 0;
        }

        public string Name
        {
            get { return _name; }
            set
            {
                if (value.Length > 200)
                    throw new Exception(Literals.ExceptionMessages.NameLength);

                _name = value;
            }
        }

        public string Country
        {
            get { return _country; }
            set
            {
                if (value.Length > 75)
                    throw new Exception(Literals.ExceptionMessages.ContryLength);
            }
        }
    }
}
```

```

        foreach (var ch in value)
            if (!(char.IsLetter(ch) || ch == '-' || ch == '.' || ch == ',' || ch == ' '))
                throw new
Exception(Literals.ExceptionMessages.CountryName);

        _country = value;
    }
}

public double Cost
{
    get { return _costPerKg; }
    set
    {
        if (value < 0)
            throw new Exception(Literals.ExceptionMessages.CostEx);

        _costPerKg = value;
    }
}

public int MinTemperature
{
    get { return _minTemperature; }
    set
    {
        if (value < -101)
            throw new Exception(Literals.ExceptionMessages.MinTlow);

        else if (value > 100)
            throw new Exception(Literals.ExceptionMessages.MinThigh);

        else if (MaxTemperature != 101 && value > MaxTemperature)
            throw new Exception(Literals.ExceptionMessages.MinMax);

        _minTemperature = value;
    }
}

public int MaxTemperature
{
    get { return _maxTemperature; }
    set
    {
        if (value < -100)
            throw new Exception(Literals.ExceptionMessages.MaxTlow);

        else if (value > 101)
            throw new Exception(Literals.ExceptionMessages.MaxThigh);
    }
}

```

```
        else if (MinTemperature != -101 && value < MinTemperature)
            throw new Exception(Literals.ExceptionMessages.MaxMin);

        _maxTemperature = value;
    }
}

public uint ShelfLife
{
    get { return _shelfLife; }
    set
    {
        if (value > 120)
            throw new Exception(Literals.ExceptionMessages.ShelfLifeEx);

        _shelfLife = value;
    }
}

public int Calories
{
    get { return _calories; }
    set
    {
        if (value < -1)
            throw new Exception(Literals.ExceptionMessages.CaloriesLessZero);

        _calories = value;
    }
}

public abstract bool IsOverdue
{
    get;
}

public abstract string GetKind
{
    get;
}
}
```

## IceCream.cs

```
using System;
using CommonWorkingLibrary;

namespace SweetLibrary
{
    public class IceCream : Sweets // Производный класс "Мороженое"
    {
        private string _date;
        private string _taste;

        public IceCream()
        {
            _date = String.Empty;
            _taste = String.Empty;
        }

        public string Taste
        {
            get { return _taste; }
            set
            {
                if (value.Length > 200)
                    throw new Exception(Literals.ExceptionMessages.TasteEx);
                _taste = value;
            }
        }

        public string Date
        {
            get
            {
                return String.IsNullOrEmpty(_date) ? String.Empty : _date;
            }
            set
            {
                if (String.IsNullOrEmpty(value))
                    _date = String.Empty;
                else
                {
                    DateTime temp;
                    if (!DateTime.TryParse(value, out temp) || value.Length !=
Literals.Controls.DateFormat.Length)
                        throw new
Exception(Literals.ExceptionMessages.CantMakeDate);
                    else
                        _date = temp.ToString(Literals.Controls.DateFormat);
                }
            }
        }
    }
}
```

```

    }

    public override string GetKind
    {
        get { return Literals.Specific.IceCream; }
    }

    public override bool IsOverdue
    {
        get
        {
            if (String.IsNullOrEmpty(Date) || ShelfLife == 0)
                return false;

            int y = DateTime.Today.Year - int.Parse(Date.Substring(3));
            int m = Math.Abs(DateTime.Today.Month -
int.Parse(Date.Substring(0,2)));
            m = Math.Abs((y * 12) - m);
            if (m >= ShelfLife)
                return true;

            else return false;
        }
    }

    public override string ToString()
    {
        if (String.IsNullOrEmpty(Name))
            return Literals.Specific.IceCream;
        else
            return String.Format("{0} «{1}»", Literals.Specific.IceCream,
Name);
    }
}

```

## DriedApricots.cs

```
using System;
using CommonWorkingLibrary;

namespace SweetLibrary
{
    public class DriedApricots : Sweets // Производный класс "Курара"
    {
        private string _date;
        private string _sort;

        public DriedApricots()
        {
            _date = String.Empty;
            _sort = String.Empty;
        }

        public string Sort
        {
            get { return _sort; }
            set
            {
                if (value.Length > 200)
                    throw new Exception(Literals.ExceptionMessages.SortEx);
                _sort = value;
            }
        }

        public string Date
        {
            get
            {
                return String.IsNullOrEmpty(_date) ? String.Empty : _date;
            }
            set
            {
                if (String.IsNullOrEmpty(value))
                    _date = String.Empty;
                else
                {
                    DateTime temp;
                    if (!DateTime.TryParse(value, out temp) || value.Length !=
Literals.Controls.DateFormat.Length)
                        throw new
Exception(Literals.ExceptionMessages.CantMakeDate);
                    else
                        _date = temp.ToString(Literals.Controls.DateFormat);
                }
            }
        }
    }
}
```



```
    }

    public override string GetKind
    {
        get { return Literals.Specific.DriedApricots; }
    }

    public override bool IsOverdue
    {
        get
        {
            if (String.IsNullOrEmpty(Date) || ShelfLife == 0)
                return false;

            int y = DateTime.Today.Year - int.Parse(Date.Substring(3));
            int m = Math.Abs(DateTime.Today.Month - int.Parse(Date.Substring(0,
2)))));

            m = Math.Abs((y * 12) - m);
            if (m >= ShelfLife)
                return true;

            else return false;
        }
    }

    public override string ToString()
    {
        if (String.IsNullOrEmpty(Name))
            return Literals.Specific.DriedApricots;
        else
            return String.Format("{0} «{1}»", Literals.Specific.DriedApricots,
Name);
    }
}
```

**Methods.cs**

```

using System;
using System.Text;
using System.Windows.Forms;
using System.Collections.Generic;
using System.IO;
using CommonWorkingLibrary;
using SweetLibrary;

namespace SweetForms
{
    public static class Methods
    {
        private static int padRightValue =
MaxLength(Literals.Specific.ParametersArray) + 6;

        private static int MaxLength(string[] array)
        {
            int i = 0;
            foreach (var str in array)
                if (str.Length > i)
                    i = str.Length;
            return i;
        }

        public static void RefreshListBox(ListBox listbox, List<Sweets> list)
        {
            listbox.DataSource = null;
            listbox.DataSource = list;
        }

        public static void Save(string path, List<Sweets> list, bool isItExport)
        {
            StreamWriter writer = new StreamWriter(path, isItExport,
Encoding.Unicode);

            foreach (var item in list)
            {
                writer.WriteLine(Literals.Specific.Title(item.GetKind));
                writer.WriteLine(Literals.Specific.Separator);

                if (!String.IsNullOrEmpty(item.Name))
                    writer.WriteLine((Literals.Specific.Name +
':').PadRight(padRightValue) + item.Name);

                if (!String.IsNullOrEmpty(item.Country))
                    writer.WriteLine((Literals.Specific.Country +
':').PadRight(padRightValue) + item.Country);

                if(item.Cost > 0)

```

```

        writer.WriteLine((Literals.Specific.Cost
':').PadRight(padRightValue) + item.Cost.ToString("f2"));

        if(item.MinTemperature > -101)
            writer.WriteLine((Literals.Specific.MinTemperature
':').PadRight(padRightValue) + item.MinTemperature);

        if (item.MaxTemperature < 101)
            writer.WriteLine((Literals.Specific.MaxTemperature
':').PadRight(padRightValue) + item.MaxTemperature);

        if(item.Calories > -1)
            writer.WriteLine((Literals.Specific.Calories
':').PadRight(padRightValue) + item.Calories);

        if (item.ShelfLife != 0)
            writer.WriteLine((Literals.Specific.ShelfLife
':').PadRight(padRightValue) + item.ShelfLife);

        if (item is IceCream &&
!String.IsNullOrEmpty(((IceCream)item).Date))
            writer.WriteLine((Literals.Specific.Date
':').PadRight(padRightValue) + ((IceCream)item).Date);

        if (item is DriedApricots &&
!String.IsNullOrEmpty(((DriedApricots)item).Date))
            writer.WriteLine((Literals.Specific.Date
':').PadRight(padRightValue) + ((DriedApricots)item).Date);

        if (item is IceCream &&
!String.IsNullOrEmpty(((IceCream)item).Taste))
            writer.WriteLine((Literals.Specific.Taste
':').PadRight(padRightValue) + ((IceCream)item).Taste);

        if (item is DriedApricots &&
!String.IsNullOrEmpty(((DriedApricots)item).Sort))
            writer.WriteLine((Literals.Specific.Sort
':').PadRight(padRightValue) + ((DriedApricots)item).Sort);

        writer.WriteLine();
    }

    writer.Flush();
    writer.Close();
}

public static void Open(string path, List<Sweets> list, bool isItImport)
{
    StreamReader reader = new StreamReader(path, Encoding.Unicode);
    int i;
    bool everythingIsOk = true;

```

```

        if (isItImport)
            i = list.Count - 1;

        else
        {
            list.Clear();
            i = -1;
        }

        while (true)
        {
            string currentLine = reader.ReadLine();

            if (currentLine == null) break;

            else
                if
                (currentLine.Contains(Literals.Specific.Title(Literals.Specific.IceCream)))
                {
                    list.Add(new IceCream());
                    i++;
                }

            else
                if
                (currentLine.Contains(Literals.Specific.Title(Literals.Specific.DriedApricots)))
                {
                    list.Add(new DriedApricots());
                    i++;
                }

            else
                try
                {
                    if (currentLine.Contains(Literals.Specific.Name + ':'))
                        list[i].Name =
currentLine.Substring(currentLine.IndexOf(':') + 1).Trim();

                    else if (currentLine.Contains(Literals.Specific.Country +
':'))
                        list[i].Country =
currentLine.Substring(currentLine.IndexOf(':') + 1).Trim();

                    else if (currentLine.Contains(Literals.Specific.Cost +
':'))
                        list[i].Cost =
double.Parse(currentLine.Substring(currentLine.IndexOf(':') + 1).Trim());

                    else
                        if
                        (currentLine.Contains(Literals.Specific.MinTemperature + ':'))
                        {

```

```

                                int                checkTemp                =
int.Parse(currentLine.Substring(currentLine.IndexOf(':') + 1).Trim());
                                if (checkTemp > -101)
                                    list[i].MinTemperature = checkTemp;
                                else everythingIsOk = false;
                                }

                                else                                if
(currentLine.Contains(Literals.Specific.MaxTemperature + ':'))
                                {
                                    int                checkTemp                =
int.Parse(currentLine.Substring(currentLine.IndexOf(':') + 1).Trim());
                                    if (checkTemp < 101)
                                        list[i].MaxTemperature = checkTemp;
                                    else everythingIsOk = false;
                                }

                                else if (currentLine.Contains(Literals.Specific.Calories +
':'))
                                {
                                    int                checkTemp                =
int.Parse(currentLine.Substring(currentLine.IndexOf(':') + 1).Trim());
                                    if (checkTemp > -1)
                                        list[i].Calories = checkTemp;
                                    else everythingIsOk = false;
                                }

                                else if (currentLine.Contains(Literals.Specific.ShelfLife +
':'))
                                    list[i].ShelfLife                =
uint.Parse(currentLine.Substring(currentLine.IndexOf(':') + 1).Trim());

                                else if (currentLine.Contains(Literals.Specific.Date +
':'))
                                {
                                    if (list[i] is IceCream)
                                        ((IceCream)list[i]).Date                =
currentLine.Substring(currentLine.IndexOf(':') + 1).Trim();
                                    else if (list[i] is DriedApricots)
                                        ((DriedApricots)list[i]).Date                =
currentLine.Substring(currentLine.IndexOf(':') + 1).Trim();
                                }

                                else if (currentLine.Contains(Literals.Specific.Taste +
':'))
                                    ((IceCream)list[i]).Taste                =
currentLine.Substring(currentLine.IndexOf(':') + 1).Trim();

                                else if (currentLine.Contains(Literals.Specific.Sort +
':'))

```

```
                ((DriedApricots)list[i]).Sort
currentLine.Substring(currentLine.IndexOf(':') + 1).Trim();
            }
            catch { everythingIsOk = false; }
        }

        reader.Close();
        if (!everythingIsOk)
            MessageBox.Show(Literals.ExceptionMessages.OpenFileError,
Literals.Controls.Error, MessageBoxButtons.OK);
    }
}
```

## **6. СПИСОК ЛИТЕРАТУРЫ**

<http://msdn.microsoft.com/>

**Герберт Шилдт - C# 4.0. Полное руководство.**