**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**КАФДЕРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

**КУРСОВА РОБОТА**

**з дисципліни “Об’єктно-орієнтоване програмування”**

**на тему:** “Склад продовольчих та непродовольчих товарів”

Студента групи ПЗ-21

спеціальності 6.121

“Інженерія програмного забезпечення”

Мартинюка Н.В.

Керівник: доцент кафедри ПЗ,

к.ф.м.н., доцент Дяконюк Л.М.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_Оцінка:  ECTS \_\_\_\_\_

 Члени комісії          \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Львів — 2020

**Зміст**

**1.** [**Завдання**](#Завдання)**……………………………………………………………………..3**

**2.** [**Алгоритми розв’язку задачі**](#АлгоритмиРЗ)**………………………………………………5**

**3.** [**Діаграми**](#UML)**…………………………………………………………………….10**

3.1. [Діаграма прецедентів](#Прецеденти)………………………………………………10

3.2. [Діаграма класів](#Класи)……………………………………………………..11

3.3. [Діаграма послідовності](#Послідовності)…………………………………………….12

**4.** [**Код програми**](#Код)**……………………………………………………………….13**

4.1. [Goods.cs](#Goods)…...………………....………………………………….…...13

4.2. [Groceries.cs](#Groceries)…………………………………………………………..13

4.3. [NonGroceries.cs](#NonGroceries)…………………………………………………...…13

4.4. [DestinationGroup.cs](#Destination)………………………………………………….14

4.5. [Dimensions.cs](#Dimensions)………………………………………………………...14

4.6. [CompareItems.cs](#Compare)……………………………………………………...15

4.7. [Storage.cs](#Storage)……………………………………………………………..16

4.8. [AddForm.cs](#AddForm)……………………………………………………....…..17

4.9. [FilterForm.cs](#FilterForm)………………………………………………………….23

4.10. [MainForm.cs](#MainForm)………………………………………………………...35

4.11. [Program.cs](#Program)…………………………………………………………..46

**5.** [**Протокол роботи програми**](#Протокол)**………………………………………………..47**

**6.** [**Інструкція користувача**](#Інструкція)**……………………………………………………55**

**7.** [**Структура файлу вхідних даних**](#Файл)**……………………………………….....62**

**8. [Виняткові ситуації](#Винятки)………………………………………………………….63**

[**Висновки**](#Висновки)**………………………………………………………………………..64**

[**Список використаних джерел**](#Література)**………………………………………………..64**

**1.** **Завдання**

до курсової роботи з дисципліни “Об’єктно-орієнтоване програмування”

студента групи ПЗ-21 Мартинюка Назара

**Тема: “Склад продовольчих та непродовольчих товарів”**

Варіант №3

Створити програму, що створює модель складу продовольчих та непродовольчих товарів.

Програма передбачає роботу з продовольчими товарами, які мають назву, вагу, вартість, кількість, дату створення, термін придатності, виробника. Непродовольчі товари характеризуються назвою, вартістю, кількістю, виробником, габаритними розмірами, групою призначення (одяг, техніка, меблі, посуд та ін.). Для всіх товарів існує функція зміни вартості, яка передбачає зниження вартості у випадку, якщо термін придатності вичерпаний на 80% для продовольчих товарів, додаткову надбавку за перевищення норм габаритів – у випадку непродовольчих товарів.

Потрібно створити наступні сервіси:

1. Наповнення складу конкретними товарами.

2. Розподіл товарів за різними групами.

3. Пошук товарів за різними критеріями.

4. Зміну вартості всіх товарів.

5. Сортування інформації за різними критеріями.

6. Долучення та вилучення нових товарів.

7. Визначення унікальних товарів, які не мають повторів з зазначенням критеріїв порівняння на еквівалентність двох товарів.

8. Визначення сумарної кількості товарів заданої назви.

9. Визначення товарів заданої категорії, обмежених ціновим діапазоном.

**Зміст завдання та календарний план його виконання**

|  |  |  |
| --- | --- | --- |
| **№ з/п** | **Зміст завдання** | **Дата** |
| 1 | Здійснити аналiтичний огляд лiтератури за заданою темою та обгрунтувати вибір інструментальних засобів реалізації. | 15.10.2020  - 24.10.2020 |
| 2 | Побудова UML діаграм | 25.10.2020  - 29.10.2020 |
| 3 | Розробка алгоритмів реалізації | 30.10.2020  - 03.11.2020 |
| 4 | Реалізація завдання (кодування) | 04.11.2020  - 24.11.2020 |
| 5 | Формування інструкції користувача | 25.11.2020  - 27.11.2020 |
| 6 | Оформлення звіту до курсової роботи згідно з вимогами Міжнародних стандартів, дотримуючись такої структури:  ·       зміст;  ·        алгоритм розв‘язку задачі у покроковому представленні;  ·        діаграми UML [клас](http://vns.lpnu.ua/mod/glossary/showentry.php?eid=54317&displayformat=dictionary)ів, прецедентів, послідовності виконання;  ·        код розробленої програми з коментарями;  ·        протокол роботи програми для кожного пункту завдання  ·        інструкція користувача та системні вимоги;  ·        опис виняткових ситуацій;  ·        структура файлу вхідних даних;  ·        висновки;  ·        список використаних джерел. | 28.11.2020  - 30.11.2020 |

Завдання прийнято до виконання: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Прізвище І.П.)

( пiдпис студента )

Керівник роботи: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Дяконюк Л. М./

**2.** **Алгоритми розв’язку задачі**

2.1. Алгоритм роботи програми (повний функціонал)

Алгоритм R

R1. Відобразити порожню таблицю товарів

R2. Переглянути файл для додавання тренажерів натисканням на кнопку з піктограмою файлу

R3. Додавати товари на склад натисканням на кнопку з піктограмою файлу

R4. Відобразити таблицю доданих товарів

R5. Перейти в меню додавання товару вручну натисканням на кнопку з піктограмою плюса

R6. Додати товар до списку заповнивши всі поля згідно інструкцій програми натисканням кнопки з піктограмою галочки

R7. Відобразити таблицю доданих товарів

R8. Натискаючи лівою кнопкою миші на назву товару, відобразити повну інформацію про товар

R9. Виділяючи декілька товарів одночасно відобразити повну інформацію про останній виділений товар

R10. Натискаючи лівою кнопкою миші на назву товару, відобразити кількість товарів з такою самою назвою на складі

R11. Натискаючи на кнопку з піктограмою дискети, зберегти всю інформацію в текстовий файл

R12. Натискаючи в списку на потрібну шапку стовбця, відсортувати список відносно заданого критерія

R13. Обравши один або декілька товарів і натиснувши на кнопку з піктограмою смітника, видалити товар з складу

R14. Натискаючи на кнопку з піктограмою фільтру, заповнивши потрібні користувачеві для фільтрування поля та натиснувши на кнопку з піктограмою лупи, вивести список відфільтрованих товарів

R15. Натискаючи на кнопку з піктограмою фільтру, заповнивши всі поля та натиснувши на кнопку з піктограмою унікальності, вивести список товарів, критерії яких, не збігаються зі всіма полями

R16. Натиснувши на кнопку з піктограмою оновлення, оновити інформацію в програмі, зменшити ціни на продовольчі товари, якщо це потрібно, видалити продовольчі товари, якщо це потрібно.

R17. Завершити роботу з програмою.

R18. Кінець.

2.2. Задача наповнення складу конкретними товарами

Алгоритм N

N1. Створити об’єкт типу Groceries або NonGroceries(в залежності від вибору користувача)

N2. Зчитати назву товару і встановити її в поле name

N3. Зчитати виробника товару і встановити її в поле producer

N4. Зчитати ціну товару і встановити її в поле price

N5. Зчитати кількість товару і встановити її в поле quantity

N6. Якщо користувач обрав продовольчі товари, то перейти до пункту N7, в іншому випадку до пункту N11

N7. Зчитати вагу товару і встановити її в поле weight

N8. Зчитати дату створення товару і встановити її в поле dateCreation

N9. Зчитати термін придатності товару і встановити його в поле dateExpiration

N10. Додати об’єкт Groceries в кінець списку і перейти до пункту N14

N11. Зчитати габарити товару і встановити їх в поле dimensions

N12. Зчитати групу призначення товару і встановити його в поле group

N13. Додати об’єкт NonGroceries в кінець списку

N14. Кінець

2.3. Задача розподілу товарів за різними групами

Алгоритм M

M1. Зчитати інформацію про вибір користувача продовольчих або непродовольчих товарів

M2. Перевірити чи потрапляє елемент списку під критерій пошуку товару, якщо потрапляє, то вивести його в список

M3. Перейти до пункту M2, але з наступним елементом списку, якщо список не досягнув кінця

M4. Кінець

2.4. Задача пошуку товарів за різними критеріями

Алгоритм I

I1. Зчитати всі критерії, які ввів користувач

I2. Перевірити елемент списку, чи співпадають його поля з критеріями пошуку користувача. Якщо так, то вивести

І3. Перейти до пункту І2, але з наступним елементом списку, якщо список не досягнув кінця

І4. Кінець

2.5. Задача зміни вартості всіх товарів

Алгоритм А

А1. Зчитати нову ціну товару

А2. Знайти обраний товар в списку

А3. Змінити його вартість

А4. Змінити його вартість в виведеном списку

А5. Кінець

2.6. Задача сортування інформації за різними критеріями

Алгоритм С

С1. Зчитати інформацію критерія сортування

С2. Відсортувати список відносно критерія сортування

С3. Вивести відсортований список

С4. Кінець

2.7. Задача долучення та вилучення нових товарів

Алгоритм Z

Z1. Якщо потрібно вилучити, то перейти до пункту Z3

Z2. Виконати алгоритм N та перейти до пункту Z6

Z3. Знайти обраний товар в списку

Z4. За допомогою існуючих функцій вилучити його зі списку

Z5. Вивести новий список

Z6. Кінець

2.8. Задача визначення унікальних товарів, які не мають повторів з зазначенням критеріїв порівняння на еквівалентність двох товарів.

Алгоритм T

T1. Зчитати всю інформацію введену користувачем

Т2. Перевірити елемент списку, чи має він хоч якісь ідентичні поля відносно інформації користувача. Якщо ні, то вивести цей елемент

Т3. Перейти до пункту Т2, але з наступним елементом списку, якщо список не досягнув кінця

Т4. Кінець

2.9. Задача визначення сумарної кількості товарів заданої назви

Алгоритм О

О1. Знайти обраний елемент в списку

О2. Перевірити елемент списку, чи збігаються імена з обраним елементом списку

О3. Перейти до пункту О2, але з наступним елементом списку, якшо список не досягнув кінця

О4. Вивести кількість товарів з однаковими іменами

О5. Кінець

2.10 Задача визначення товарів заданої категорії, обмежених ціновим діапазоном

Алгоритм В

В1. Зчитати інформацію написану користувачем

В2. Перевірити елемент списку, чи входить його ціна в цей діапазон. Якщо так, то вивести цей товар

В3. Перейти до пункту В2, але з наступним елементом списку, якшо список не досягнув кінця

В4. Кінець

2.11 Задача зчитування товарів з файлу

Алгоритм Е

Е1. Відкрити вказаний користувачем файл

Е2. Зчитати одну стрічку з файлу. Якщо це Non-Grocerie то перейти до пунтку Е13

Е3. Створити об’єкт класу Groceries

E4. Зчитати стрічку з іменем і записати її в поле name

Е5. Зчитати стрічку з виробником і записати її в поле producer

E6. Зчитати стрічку з ціною і записати її в поле price

E7. Зчитати стрічку з кількістю і записати її в поле quantity

E8. Зчитати стрічку з вагою і записати її в поле weight

E9. Зчитати стрічку з датою створення і записати її в поле dateCreation

E10. Зчитати стрічку з терміном придатності і записати її в поле dateExpiration

E11. Додати об’єкт класу Groceries до списку

Е12. Перейти до пункту Е2, якщо це був не кінець файлу. В іншому випадку перейти до пункту E22

Е13. Створити об’єкт класу NonGroceries

E14. Зчитати стрічку з іменем і записати її в поле name

E15. Зчитати стрічку з виробником і записати її в поле producer

E16. Зчитати стрічку з ціною і записати її в поле price

E17. Зчитати стрічку з кількістю і записати її в поле quantity

E18. Зчитати три стрічки з габаритами і записати їх в поле dimensions

E19. Зчитати стрічку з групою призначення і записати її в поле group

E20. Додати об’єкт класу NonGroceries до списку

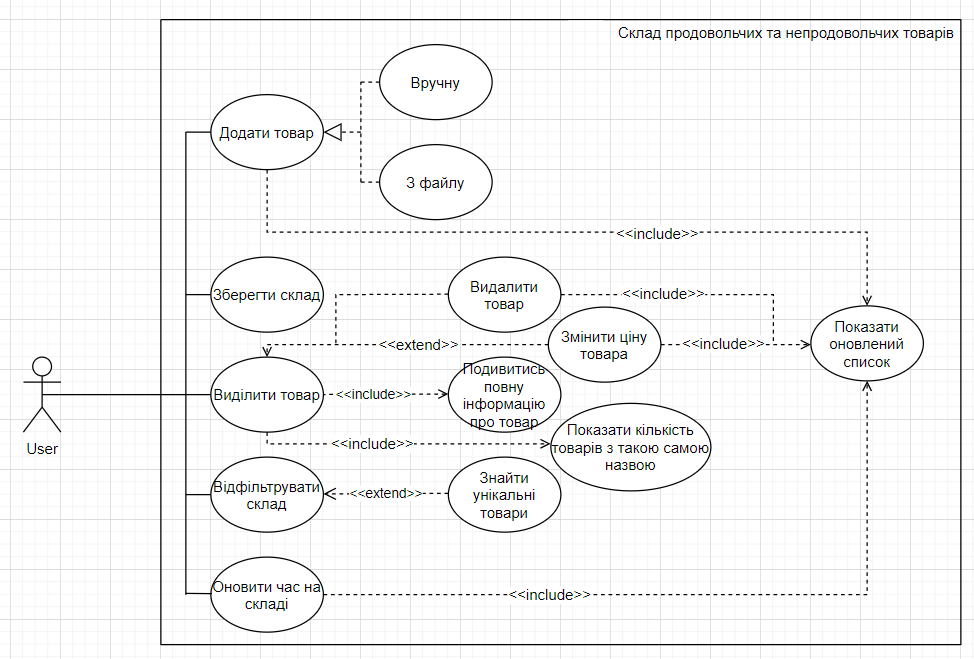
E21. Перейти до пункту E12

E22. Вивести список на екран

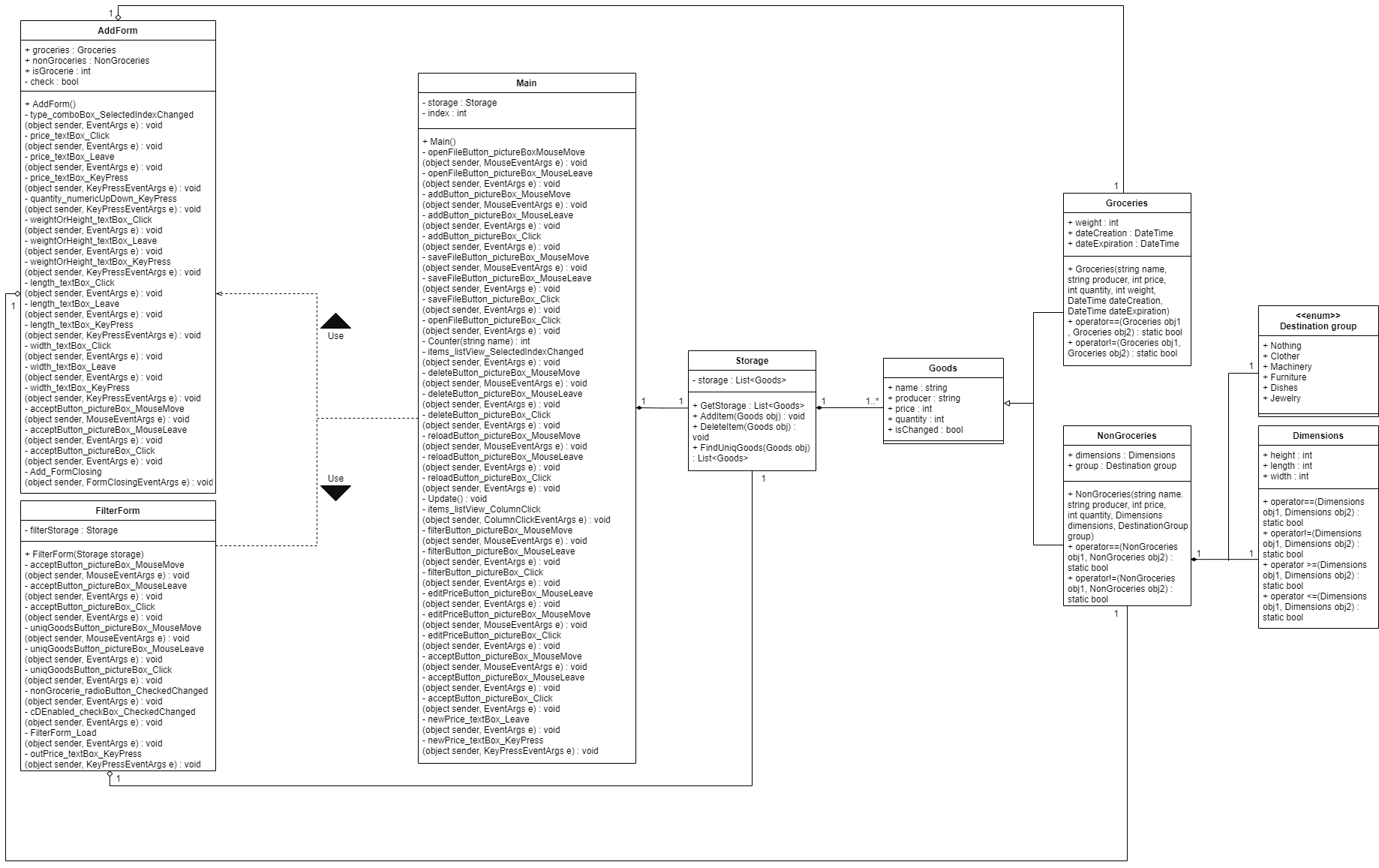
E23. Кінець

**3.** **Діаграми UML**

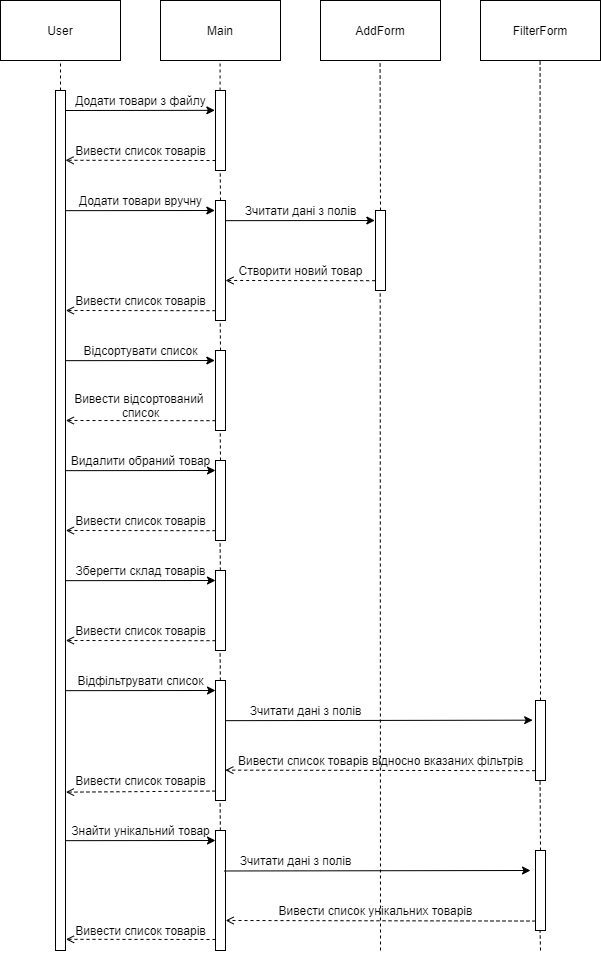
3.1. Діаграма прецедентів



3.2. Діаграма класів



3.3. Діаграма послідовності виконання



**4.** **Код програми**

4.1. Goods.cs

namespace Курсова

{

public abstract class Goods

{

public string name { get; set; }

public string producer { get; set; }

public int price { get; set; }

public int quantity { get; set; } //Кількість

public bool isChanged { get; set; }

}

}

4.2. Groceries.cs

using System;

namespace Курсова

{

public class Groceries : Goods

{

public int weight { get; }

public DateTime dateCreation { get; }

public DateTime dateExpiration { get; }

public Groceries(string name, string producer, int price, int quantity, int weight, DateTime dateCreation, DateTime dateExpiration)

{

this.name = name;

this.producer = producer;

this.price = price;

this.quantity = quantity;

this.weight = weight;

this.dateCreation = dateCreation;

this.dateExpiration = dateExpiration;

isChanged = false;

}

public static bool operator ==(Groceries obj1, Groceries obj2)

{

if (obj1.name == obj2.name && obj1.producer == obj2.producer && obj1.price == obj2.price

&& obj1.weight == obj2.weight && obj1.dateCreation == obj2.dateCreation

&& obj1.dateExpiration == obj2.dateExpiration)

return true;

else

return false;

}

public static bool operator !=(Groceries obj1, Groceries obj2) => !(obj1 == obj2);

}

}

4.3. NonGroceries.cs

namespace Курсова

{

public class NonGroceries : Goods

{

public Dimensions dimensions { get; set; }

public DestinationGroup group { get; set; }

public NonGroceries(string name, string producer, int price, int quantity, Dimensions dimensions, DestinationGroup group)

{

this.name = name;

this.producer = producer;

this.price = price;

this.quantity = quantity;

this.dimensions = dimensions;

this.group = group;

isChanged = false;

}

public static bool operator ==(NonGroceries obj1, NonGroceries obj2)

{

if (obj1.name == obj2.name && obj1.producer == obj2.producer && obj1.price == obj2.price

&& obj1.dimensions == obj2.dimensions && obj1.group == obj2.group)

return true;

else

return false;

}

public static bool operator !=(NonGroceries obj1, NonGroceries obj2) => !(obj1 == obj2);

}

}

4.4. DestinationGroup.cs

namespace Курсова

{

public enum DestinationGroup

{

Nothing,

Clothes,

Machinery,

Furniture,

Dishes,

Jewelry

}

}

4.5. Dimensions.cs

namespace Курсова

{

public struct Dimensions

{

public int height;

public int length;

public int width;

public static bool operator ==(Dimensions obj1, Dimensions obj2)

{

if (obj1.height == obj2.height && obj1.length == obj2.length && obj1.width == obj2.width)

return true;

else

return false;

}

public static bool operator !=(Dimensions obj1, Dimensions obj2) => !(obj1 == obj2);

public static bool operator >=(Dimensions obj1, Dimensions obj2)

{

if (obj1.height >= obj2.height && obj1.length >= obj2.length && obj1.width >= obj2.width)

return true;

else

return false;

}

public static bool operator <=(Dimensions obj1, Dimensions obj2)

{

if (obj1.height <= obj2.height && obj1.length <= obj2.length && obj1.width <= obj2.width)

return true;

else

return false;

}

}

}

4.6. CompareItem.cs

using System.Collections;

using System.Windows.Forms;

namespace Курсова

{

internal class CompareItems : IComparer

{

private int column;

private SortOrder sorting;

public CompareItems(int column, SortOrder sorting)

{

this.column = column;

this.sorting = sorting;

}

public int Compare(object x, object y)

{

ListViewItem one = x as ListViewItem;

ListViewItem two = y as ListViewItem;

int mult = sorting == SortOrder.Ascending ? 1 : -1;

if (column == 0)

{

return mult \* one.Text.CompareTo(two.Text);

}

else if (column == 1)

{

return mult \* one.SubItems[1].Text.CompareTo(two.SubItems[1].Text);

}

else if (column == 2)

{

return mult \* one.SubItems[2].Text.CompareTo(two.SubItems[2].Text);

}

else if (column == 3)

{

int price1, price2;

price1 = int.Parse(one.SubItems[3].Text);

price2 = int.Parse(two.SubItems[3].Text);

return mult \* price1.CompareTo(price2);

}

else

{

int quantity1, quantity2;

quantity1 = int.Parse(one.SubItems[4].Text);

quantity2 = int.Parse(two.SubItems[4].Text);

return mult \* quantity1.CompareTo(quantity2);

}

}

}

}

4.7. Storage.cs

using System;

using System.Collections.Generic;

namespace Курсова

{

public class Storage

{

private List<Goods> storage = new List<Goods>();

public List<Goods> GetStorage { get => storage; }

public void AddItem(Goods obj)

{

storage.Add(obj);

}

public void DeleteItem(Goods obj)

{

storage.Remove(obj);

}

public List<Goods> FindUniqGoods(Goods obj)

{

List<Goods> res = new List<Goods>();

if(obj is Groceries)

{

Groceries newObj = obj as Groceries;

foreach (Goods item in storage)

{

Groceries tmp = item as Groceries;

if (tmp != newObj)

res.Add(tmp);

}

}

else

{

NonGroceries newObj = obj as NonGroceries;

foreach (Goods item in storage)

{

NonGroceries tmp = item as NonGroceries;

if (tmp != newObj)

res.Add(tmp);

}

}

return res;

}

}

}

4.8. AddForm.cs

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Курсова

{

public partial class AddForm : Form

{

public Groceries groceries;

public NonGroceries nonGroceries;

public int isGrocerie;

bool check = false;

public AddForm()

{

InitializeComponent();

dateCreation\_dateTimePicker.MaxDate = DateTime.Now;

dateCreation\_dateTimePicker.Value = DateTime.Now;

dateCreation\_dateTimePicker.MinDate = DateTime.Now.AddYears(-100);

dateExpiration\_dateTimePicker.MaxDate = DateTime.Now.AddYears(100);

dateExpiration\_dateTimePicker.MinDate = DateTime.Today.AddDays(1);

}

private void type\_comboBox\_SelectedIndexChanged(object sender, EventArgs e)

{

acceptButton\_pictureBox.Visible = true;

name\_textBox.ReadOnly = false;

producer\_textBox.ReadOnly = false;

price\_textBox.ReadOnly = false;

price\_textBox.Text = "0";

price\_textBox.ForeColor = SystemColors.InactiveCaption;

quantity\_label.Visible = true;

quantity\_numericUpDown.Visible = true;

changable1\_label.Visible = true;

weightOrHeight\_textBox.Visible = true;

changable2\_label.Visible = true;

weightOrHeight\_textBox.ForeColor = SystemColors.InactiveCaption;

if (type\_comboBox.SelectedItem.ToString() == "Grocerie")

{

isGrocerie = 1;

changable1\_label.Text = "WEIGHT(KG):";

weightOrHeight\_textBox.Text = "0";

changable2\_label.Text = "CREATION DATE:";

length\_textBox.Visible = false;

width\_textBox.Visible = false;

destinationGroup\_comboBox.Location = new Point(592, 292);

destinationGroup\_comboBox.Visible = false;

dateCreation\_dateTimePicker.Visible = true;

dateCreation\_dateTimePicker.Location = new Point(179, 239);

expirationDate\_label.Visible = true;

dateExpiration\_dateTimePicker.Visible = true;

}

else

{

isGrocerie = 2;

changable1\_label.Text = "DIMENSIONS:";

weightOrHeight\_textBox.Text = "Height";

length\_textBox.Visible = true;

length\_textBox.ForeColor = SystemColors.InactiveCaption;

width\_textBox.Visible = true;

width\_textBox.ForeColor = SystemColors.InactiveCaption;

changable2\_label.Text = "DESTINATION GROUP:";

destinationGroup\_comboBox.Location = new Point(200, 236);

destinationGroup\_comboBox.Visible = true;

dateCreation\_dateTimePicker.Visible = false;

dateCreation\_dateTimePicker.Location = new Point(592, 292);

expirationDate\_label.Visible = false;

dateExpiration\_dateTimePicker.Visible = false;

}

}

private void price\_textBox\_Click(object sender, EventArgs e)

{

if (price\_textBox.Text == "0")

{

price\_textBox.Text = "";

price\_textBox.ForeColor = SystemColors.WindowText;

}

}

private void price\_textBox\_Leave(object sender, EventArgs e)

{

if (price\_textBox.Text == "")

{

price\_textBox.Text = "0";

price\_textBox.ForeColor = SystemColors.InactiveCaption;

}

}

private void price\_textBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Char.IsDigit(e.KeyChar) && !e.KeyChar.Equals(Convert.ToChar(Keys.Back)))

e.Handled = true;

else

e.Handled = false;

}

private void quantity\_numericUpDown\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Char.IsDigit(e.KeyChar) && !e.KeyChar.Equals(Convert.ToChar(Keys.Back)))

e.Handled = true;

else

e.Handled = false;

}

private void weightOrHeight\_textBox\_Click(object sender, EventArgs e)

{

if (weightOrHeight\_textBox.Text == "0" || weightOrHeight\_textBox.Text == "Height")

{

weightOrHeight\_textBox.Text = "";

weightOrHeight\_textBox.ForeColor = SystemColors.WindowText;

}

}

private void weightOrHeight\_textBox\_Leave(object sender, EventArgs e)

{

if (weightOrHeight\_textBox.Text == "" && type\_comboBox.SelectedItem.ToString() == "Groceries")

{

weightOrHeight\_textBox.Text = "0";

weightOrHeight\_textBox.ForeColor = SystemColors.InactiveCaption;

}

if (weightOrHeight\_textBox.Text == "" && type\_comboBox.SelectedItem.ToString() == "Non-grocerie")

{

weightOrHeight\_textBox.Text = "Height";

weightOrHeight\_textBox.ForeColor = SystemColors.InactiveCaption;

}

}

private void weightOrHeight\_textBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Char.IsDigit(e.KeyChar) && !e.KeyChar.Equals(Convert.ToChar(Keys.Back)))

e.Handled = true;

else

e.Handled = false;

}

private void length\_textBox\_Click(object sender, EventArgs e)

{

if (length\_textBox.Text == "Length")

{

length\_textBox.Text = "";

length\_textBox.ForeColor = SystemColors.WindowText;

}

}

private void length\_textBox\_Leave(object sender, EventArgs e)

{

if (length\_textBox.Text == "")

{

length\_textBox.Text = "Length";

length\_textBox.ForeColor = SystemColors.InactiveCaption;

}

}

private void length\_textBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Char.IsDigit(e.KeyChar) && !e.KeyChar.Equals(Convert.ToChar(Keys.Back)))

e.Handled = true;

else

e.Handled = false;

}

private void width\_textBox\_Click(object sender, EventArgs e)

{

if (width\_textBox.Text == "Width")

{

width\_textBox.Text = "";

width\_textBox.ForeColor = SystemColors.WindowText;

}

}

private void width\_textBox\_Leave(object sender, EventArgs e)

{

if (width\_textBox.Text == "")

{

width\_textBox.Text = "Length";

width\_textBox.ForeColor = SystemColors.InactiveCaption;

}

}

private void width\_textBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Char.IsDigit(e.KeyChar) && !e.KeyChar.Equals(Convert.ToChar(Keys.Back)))

e.Handled = true;

else

e.Handled = false;

}

private void acceptButton\_pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

if (type\_comboBox.SelectedItem.ToString() == "Grocerie" || type\_comboBox.SelectedItem.ToString() == "Non-grocerie")

acceptButton\_pictureBox.BackColor = Color.AliceBlue;

}

private void acceptButton\_pictureBox\_MouseLeave(object sender, EventArgs e)

{

acceptButton\_pictureBox.BackColor = SystemColors.ActiveCaption;

}

private void acceptButton\_pictureBox\_Click(object sender, EventArgs e)

{

try

{

if (name\_textBox.Text == "")

{

Exception exception = new Exception("name");

throw exception;

}

if (producer\_textBox.Text == "")

{

Exception exception = new Exception("producer");

throw exception;

}

if (Convert.ToUInt64(price\_textBox.Text) == 0)

{

Exception exception = new Exception("price");

throw exception;

}

if (Convert.ToUInt16(quantity\_numericUpDown.Value) == 0)

{

Exception exception = new Exception("quantity");

throw exception;

}

if (type\_comboBox.SelectedItem.ToString() == "Grocerie")

if (int.Parse(weightOrHeight\_textBox.Text) == 0)

{

Exception exception = new Exception("weight");

throw exception;

}

if (type\_comboBox.SelectedItem.ToString() == "Non-grocerie")

{

if (weightOrHeight\_textBox.Text == "Height" || Convert.ToUInt64(weightOrHeight\_textBox.Text) == 0 || length\_textBox.Text == "Length" || Convert.ToUInt64(length\_textBox.Text) == 0 || width\_textBox.Text == "Width" || Convert.ToUInt64(width\_textBox.Text) == 0)

{

Exception exception = new Exception("dimensions");

throw exception;

}

if (destinationGroup\_comboBox.SelectedIndex == -1)

{

Exception exception = new Exception("destination group");

throw exception;

}

}

string name = name\_textBox.Text;

int price = Convert.ToInt32(price\_textBox.Text);

int quantity = Convert.ToInt32(quantity\_numericUpDown.Value);

string producer = producer\_textBox.Text;

if (type\_comboBox.SelectedItem.ToString() == "Grocerie")

{

check = true;

isGrocerie = 1;

int weight = Convert.ToInt32(weightOrHeight\_textBox.Text);

DateTime dateCreation = dateCreation\_dateTimePicker.Value;

DateTime expirationDate = dateExpiration\_dateTimePicker.Value;

TimeSpan usingTime;

TimeSpan remainingTime;

double usingTimeI;

double remainingTimeI;

usingTime = expirationDate - dateCreation;

remainingTime = expirationDate - DateTime.Now;

usingTimeI = usingTime.TotalSeconds;

remainingTimeI = remainingTime.TotalSeconds;

bool isChanged;

if ((usingTimeI \* 0.2) > remainingTimeI)

{

price /= 2;

isChanged = true;

}

else

isChanged = false;

groceries = new Groceries(name, producer, price, quantity, weight, dateCreation, expirationDate);

groceries.isChanged = isChanged;

this.Close();

}

else

{

check = true;

isGrocerie = 2;

Dimensions dimensions = new Dimensions();

dimensions.height = Convert.ToInt32(weightOrHeight\_textBox.Text);

dimensions.length = Convert.ToInt32(length\_textBox.Text);

dimensions.width = Convert.ToInt32(width\_textBox.Text);

DestinationGroup destinationGroup = new DestinationGroup();

switch (destinationGroup\_comboBox.SelectedIndex)

{

case 0:

destinationGroup = DestinationGroup.Clothes;

break;

case 1:

destinationGroup = DestinationGroup.Machinery;

break;

case 2:

destinationGroup = DestinationGroup.Furniture;

break;

case 3:

destinationGroup = DestinationGroup.Dishes;

break;

case 4:

destinationGroup = DestinationGroup.Jewelry;

break;

default:

destinationGroup = DestinationGroup.Nothing;

break;

}

bool isChanged;

if ((dimensions.height + dimensions.length + dimensions.width) > 600)

{

price \*= 120;

price /= 100;

isChanged = true;

}

else

isChanged = false;

nonGroceries = new NonGroceries(name, producer, price, quantity, dimensions, destinationGroup);

nonGroceries.isChanged = isChanged;

this.Close();

}

}

catch (Exception text)

{

string message = "You made wrong input in ";

message += text.Message;

MessageBox.Show(message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void Add\_FormClosing(object sender, FormClosingEventArgs e)

{

if (check == false)

isGrocerie = 0;

}

}

}

4.9. FilterForm.cs

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace Курсова

{

public partial class FilterForm : Form

{

Storage filterStorage;

public FilterForm(Storage storage)

{

InitializeComponent();

filterStorage = storage;

}

private void acceptButton\_pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

acceptButton\_pictureBox.BorderStyle = BorderStyle.FixedSingle;

}

private void acceptButton\_pictureBox\_MouseLeave(object sender, EventArgs e)

{

acceptButton\_pictureBox.BorderStyle = BorderStyle.None;

}

private void acceptButton\_pictureBox\_Click(object sender, EventArgs e)

{

items\_listView.Items.Clear();

List<Goods> filterGoods = new List<Goods>();

bool check = false;

if (name\_textBox.Text != String.Empty)

{

check = true;

foreach (Goods item in filterStorage.GetStorage)

{

if (item.name == name\_textBox.Text)

filterGoods.Add(item);

}

}

if (producer\_textBox.Text != String.Empty)

{

if (check)

{

for (int i = filterGoods.Count; i >= 0; i--)

foreach (Goods item in filterGoods)

{

if (item.producer != producer\_textBox.Text)

{

filterGoods.Remove(item);

break;

}

}

}

else

{

check = true;

foreach (Goods item in filterStorage.GetStorage)

{

if (item.producer == producer\_textBox.Text)

filterGoods.Add(item);

}

}

}

if (outPrice\_textBox.Text != String.Empty && toPrice\_textBox.Text != String.Empty)

{

if (check)

{

for (int i = filterGoods.Count; i >= 0; i--)

foreach (Goods item in filterGoods)

{

if (item.price < int.Parse(outPrice\_textBox.Text) || item.price > int.Parse(toPrice\_textBox.Text))

{

filterGoods.Remove(item);

break;

}

}

}

else

{

check = true;

foreach (Goods item in filterStorage.GetStorage)

{

if (item.price >= int.Parse(outPrice\_textBox.Text) && item.price <= int.Parse(toPrice\_textBox.Text))

filterGoods.Add(item);

}

}

}

if (outQuantity\_textBox.Text != String.Empty && toQuantity\_textBox.Text != String.Empty)

{

if (check)

{

for (int i = filterGoods.Count; i >= 0; i--)

foreach (Goods item in filterGoods)

{

if (item.quantity < int.Parse(outQuantity\_textBox.Text) || item.quantity > int.Parse(toQuantity\_textBox.Text))

{

filterGoods.Remove(item);

break;

}

}

}

else

{

check = true;

foreach (Goods item in filterStorage.GetStorage)

{

if (item.quantity >= int.Parse(outQuantity\_textBox.Text) && item.quantity <= int.Parse(toQuantity\_textBox.Text))

filterGoods.Add(item);

}

}

}

if (grocerie\_radioButton.Checked)

{

if (outWeight\_textBox.Text != String.Empty && toWeight\_textBox.Text != String.Empty)

{

if (check)

{

for (int i = filterGoods.Count; i >= 0; i--)

foreach (Goods item in filterGoods)

{

if (item is Groceries)

{

Groceries tmp = item as Groceries;

if (tmp.weight < int.Parse(outWeight\_textBox.Text) || tmp.weight > int.Parse(toWeight\_textBox.Text))

{

filterGoods.Remove(item);

break;

}

}

else

{

filterGoods.Remove(item);

break;

}

}

}

else

{

check = true;

foreach (Goods item in filterStorage.GetStorage)

{

if (item is Groceries)

{

Groceries tmp = item as Groceries;

if (tmp.weight >= int.Parse(outWeight\_textBox.Text) && tmp.weight <= int.Parse(toWeight\_textBox.Text))

filterGoods.Add(item);

}

}

}

}

if (cDEnabled\_checkBox.Checked)

{

if (check)

{

for (int i = filterGoods.Count; i >= 0; i--)

foreach (Goods item in filterGoods)

{

if (item is Groceries)

{

Groceries tmp = item as Groceries;

if (tmp.dateCreation < outDateCreation\_dateTimePicker.Value || tmp.dateCreation > toDateCreation\_dateTimePicker.Value)

{

filterGoods.Remove(item);

break;

}

}

else

{

filterGoods.Remove(item);

break;

}

}

}

else

{

check = true;

foreach (Goods item in filterStorage.GetStorage)

{

if (item is Groceries)

{

Groceries tmp = item as Groceries;

if (tmp.dateCreation >= outDateCreation\_dateTimePicker.Value && tmp.dateCreation <= toDateCreation\_dateTimePicker.Value)

filterGoods.Add(item);

}

}

}

}

if (eDEnabled\_checkBox.Checked)

{

if (check)

{

for (int i = filterGoods.Count; i >= 0; i--)

foreach (Goods item in filterGoods)

{

if (item is Groceries)

{

Groceries tmp = item as Groceries;

if (tmp.dateExpiration < outDateExpiration\_dateTimePicker.Value || tmp.dateExpiration > toDateExpiration\_dateTimePicker.Value)

{

filterGoods.Remove(item);

break;

}

}

else

{

filterGoods.Remove(item);

break;

}

}

}

else

{

check = true;

foreach (Goods item in filterStorage.GetStorage)

{

if (item is Groceries)

{

Groceries tmp = item as Groceries;

if (tmp.dateExpiration >= outDateExpiration\_dateTimePicker.Value && tmp.dateExpiration <= toDateExpiration\_dateTimePicker.Value)

filterGoods.Add(item);

}

}

}

}

}

else

{

if (outWidth\_textBox.Text != String.Empty && toWidth\_textBox.Text != String.Empty)

{

if (check)

{

for (int i = filterGoods.Count; i >= 0; i--)

foreach (Goods item in filterGoods)

{

if (item is NonGroceries)

{

NonGroceries tmp = item as NonGroceries;

if (tmp.dimensions.width < int.Parse(outWidth\_textBox.Text) || tmp.dimensions.width > int.Parse(toWidth\_textBox.Text))

{

filterGoods.Remove(item);

break;

}

}

else

{

filterGoods.Remove(item);

break;

}

}

}

else

{

check = true;

foreach (Goods item in filterStorage.GetStorage)

{

if (item is NonGroceries)

{

NonGroceries tmp = item as NonGroceries;

if (tmp.dimensions.width >= int.Parse(outWidth\_textBox.Text) && tmp.dimensions.width <= int.Parse(toWidth\_textBox.Text))

filterGoods.Add(item);

}

}

}

}

if (outHeight\_textBox.Text != String.Empty && toHeight\_textBox.Text != String.Empty)

{

if (check)

{

for (int i = filterGoods.Count; i >= 0; i--)

foreach (Goods item in filterGoods)

{

if (item is NonGroceries)

{

NonGroceries tmp = item as NonGroceries;

if (tmp.dimensions.height < int.Parse(outHeight\_textBox.Text) || tmp.dimensions.height > int.Parse(toHeight\_textBox.Text))

{

filterGoods.Remove(item);

break;

}

}

else

{

filterGoods.Remove(item);

break;

}

}

}

else

{

check = true;

foreach (Goods item in filterStorage.GetStorage)

{

if (item is NonGroceries)

{

NonGroceries tmp = item as NonGroceries;

if (tmp.dimensions.height >= int.Parse(outHeight\_textBox.Text) && tmp.dimensions.height <= int.Parse(toHeight\_textBox.Text))

filterGoods.Add(item);

}

}

}

}

if (outLength\_textBox.Text != String.Empty && toLength\_textBox.Text != String.Empty)

{

if (check)

{

for (int i = filterGoods.Count; i >= 0; i--)

foreach (Goods item in filterGoods)

{

if (item is NonGroceries)

{

NonGroceries tmp = item as NonGroceries;

if (tmp.dimensions.length < int.Parse(outLength\_textBox.Text) || tmp.dimensions.length > int.Parse(toLength\_textBox.Text))

{

filterGoods.Remove(item);

break;

}

}

else

{

filterGoods.Remove(item);

break;

}

}

}

else

{

check = true;

foreach (Goods item in filterStorage.GetStorage)

{

if (item is NonGroceries)

{

NonGroceries tmp = item as NonGroceries;

if (tmp.dimensions.length >= int.Parse(outLength\_textBox.Text) && tmp.dimensions.length <= int.Parse(toLength\_textBox.Text))

filterGoods.Add(item);

}

}

}

}

if (group\_comboBox.SelectedIndex != -1)

{

DestinationGroup group;

switch (group\_comboBox.SelectedIndex)

{

case 0:

group = DestinationGroup.Clothes;

break;

case 1:

group = DestinationGroup.Machinery;

break;

case 2:

group = DestinationGroup.Furniture;

break;

case 3:

group = DestinationGroup.Dishes;

break;

case 4:

group = DestinationGroup.Jewelry;

break;

default:

group = DestinationGroup.Nothing;

break;

}

if (check)

{

for (int i = filterGoods.Count; i >= 0; i--)

foreach (Goods item in filterGoods)

{

if (item is NonGroceries)

{

NonGroceries tmp = item as NonGroceries;

if (tmp.group != group)

{

filterGoods.Remove(item);

break;

}

}

else

{

filterGoods.Remove(item);

break;

}

}

}

else

{

check = true;

foreach (Goods item in filterStorage.GetStorage)

{

if (item is NonGroceries)

{

NonGroceries tmp = item as NonGroceries;

if (tmp.group == group)

filterGoods.Add(item);

}

}

}

}

}

if (check)

{

foreach (Goods item in filterGoods)

{

ListViewItem listViewItem = new ListViewItem(item.name);

if (item is Groceries)

listViewItem.SubItems.Add("Grocerie");

else

listViewItem.SubItems.Add("NonGrocerie");

listViewItem.SubItems.Add(item.producer);

listViewItem.SubItems.Add(item.price.ToString());

listViewItem.SubItems.Add(item.quantity.ToString());

items\_listView.Items.Add(listViewItem);

}

}

else

{

foreach (Goods item in filterStorage.GetStorage)

{

if (grocerie\_radioButton.Checked)

{

if (item is Groceries)

{

ListViewItem listViewItem = new ListViewItem(item.name);

listViewItem.SubItems.Add("Grocerie");

listViewItem.SubItems.Add(item.price.ToString());

listViewItem.SubItems.Add(item.quantity.ToString());

items\_listView.Items.Add(listViewItem);

}

}

else

{

if (item is NonGroceries)

{

ListViewItem listViewItem = new ListViewItem(item.name);

listViewItem.SubItems.Add("NonGrocerie");

listViewItem.SubItems.Add(item.price.ToString());

listViewItem.SubItems.Add(item.quantity.ToString());

items\_listView.Items.Add(listViewItem);

}

}

}

}

}

private void uniqGoodsButton\_pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

uniqGoodsButton\_pictureBox.BorderStyle = BorderStyle.FixedSingle;

}

private void uniqGoodsButton\_pictureBox\_MouseLeave(object sender, EventArgs e)

{

uniqGoodsButton\_pictureBox.BorderStyle = BorderStyle.None;

}

private void uniqGoodsButton\_pictureBox\_Click(object sender, EventArgs e)

{

bool check = false;

if (name\_textBox.Text == String.Empty || producer\_textBox.Text == String.Empty || outPrice\_textBox.Text == String.Empty || outQuantity\_textBox.Text == String.Empty)

check = true;

List<Goods> uniqGoods;

if (grocerie\_radioButton.Checked)

{

if (outWeight\_textBox.Text == String.Empty)

check = true;

if (!cDEnabled\_checkBox.Checked || !eDEnabled\_checkBox.Checked)

check = true;

if (!check)

{

items\_listView.Items.Clear();

toPrice\_textBox.Text = outPrice\_textBox.Text;

toQuantity\_textBox.Text = outQuantity\_textBox.Text;

toWeight\_textBox.Text = outWeight\_textBox.Text;

Groceries uniq = new Groceries(name\_textBox.Text, producer\_textBox.Text, int.Parse(outPrice\_textBox.Text), int.Parse(outQuantity\_textBox.Text), int.Parse(outWeight\_textBox.Text), outDateCreation\_dateTimePicker.Value, outDateExpiration\_dateTimePicker.Value);

uniqGoods = filterStorage.FindUniqGoods(uniq);

foreach (Goods item in uniqGoods)

{

ListViewItem listViewItem = new ListViewItem(item.name);

listViewItem.SubItems.Add("Grocerie");

listViewItem.SubItems.Add(item.producer);

listViewItem.SubItems.Add(item.price.ToString());

listViewItem.SubItems.Add(item.quantity.ToString());

items\_listView.Items.Add(listViewItem);

}

}

}

else

{

if (outWidth\_textBox.Text == String.Empty || outHeight\_textBox.Text == String.Empty || outLength\_textBox.Text == String.Empty || group\_comboBox.SelectedIndex == -1)

check = true;

if (!check)

{

toWidth\_textBox.Text = outWidth\_textBox.Text;

toHeight\_textBox.Text = outHeight\_textBox.Text;

toLength\_textBox.Text = outLength\_textBox.Text;

Dimensions dimensions = new Dimensions();

dimensions.height = int.Parse(outHeight\_textBox.Text);

dimensions.length = int.Parse(outLength\_textBox.Text);

dimensions.width = int.Parse(outWidth\_textBox.Text);

DestinationGroup group;

switch (group\_comboBox.SelectedIndex)

{

case 0:

group = DestinationGroup.Clothes;

break;

case 1:

group = DestinationGroup.Machinery;

break;

case 2:

group = DestinationGroup.Furniture;

break;

case 3:

group = DestinationGroup.Dishes;

break;

case 4:

group = DestinationGroup.Jewelry;

break;

default:

group = DestinationGroup.Nothing;

break;

}

NonGroceries uniq = new NonGroceries(name\_textBox.Text, producer\_textBox.Text, int.Parse(outPrice\_textBox.Text), int.Parse(outQuantity\_textBox.Text), dimensions, group);

uniqGoods = filterStorage.FindUniqGoods(uniq);

foreach (Goods item in uniqGoods)

{

ListViewItem listViewItem = new ListViewItem(item.name);

listViewItem.SubItems.Add("NonGrocerie");

listViewItem.SubItems.Add(item.producer);

listViewItem.SubItems.Add(item.price.ToString());

listViewItem.SubItems.Add(item.quantity.ToString());

items\_listView.Items.Add(listViewItem);

}

}

}

}

private void nonGrocerie\_radioButton\_CheckedChanged(object sender, EventArgs e)

{

if (grocerie\_radioButton.Checked)

{

outDateCreation\_dateTimePicker.Value = DateTime.Now.AddDays(-1);

toDateExpiration\_dateTimePicker.Value = DateTime.Now.AddDays(2);

outWidth\_textBox.Enabled = false;

toWidth\_textBox.Enabled = false;

outHeight\_textBox.Enabled = false;

toHeight\_textBox.Enabled = false;

outLength\_textBox.Enabled = false;

toLength\_textBox.Enabled = false;

outWeight\_textBox.Enabled = true;

toWeight\_textBox.Enabled = true;

cDEnabled\_checkBox.Enabled = true;

eDEnabled\_checkBox.Enabled = true;

cDEnabled\_checkBox.Checked = false;

eDEnabled\_checkBox.Checked = false;

group\_comboBox.Enabled = false;

}

else

{

outWidth\_textBox.Enabled = true;

toWidth\_textBox.Enabled = true;

outHeight\_textBox.Enabled = true;

toHeight\_textBox.Enabled = true;

outLength\_textBox.Enabled = true;

toLength\_textBox.Enabled = true;

outWeight\_textBox.Enabled = false;

toWeight\_textBox.Enabled = false;

cDEnabled\_checkBox.Enabled = false;

eDEnabled\_checkBox.Enabled = false;

group\_comboBox.Enabled = true;

}

group\_comboBox.SelectedIndex = -1;

name\_textBox.Text = String.Empty;

producer\_textBox.Text = String.Empty;

outPrice\_textBox.Text = String.Empty;

toPrice\_textBox.Text = String.Empty;

outQuantity\_textBox.Text = String.Empty;

toQuantity\_textBox.Text = String.Empty;

outWeight\_textBox.Text = String.Empty;

toWeight\_textBox.Text = String.Empty;

outWidth\_textBox.Text = String.Empty;

toWidth\_textBox.Text = String.Empty;

outHeight\_textBox.Text = String.Empty;

toHeight\_textBox.Text = String.Empty;

outLength\_textBox.Text = String.Empty;

toLength\_textBox.Text = String.Empty;

}

private void cDEnabled\_checkBox\_CheckedChanged(object sender, EventArgs e)

{

if (cDEnabled\_checkBox.Checked)

{

outDateCreation\_dateTimePicker.Enabled = true;

toDateCreation\_dateTimePicker.Enabled = true;

}

else

{

outDateCreation\_dateTimePicker.Enabled = false;

toDateCreation\_dateTimePicker.Enabled = false;

}

if (eDEnabled\_checkBox.Checked)

{

outDateExpiration\_dateTimePicker.Enabled = true;

toDateExpiration\_dateTimePicker.Enabled = true;

}

else

{

outDateExpiration\_dateTimePicker.Enabled = false;

toDateExpiration\_dateTimePicker.Enabled = false;

}

}

private void FilterForm\_Load(object sender, EventArgs e)

{

outDateCreation\_dateTimePicker.MinDate = DateTime.Now.AddYears(-100);

outDateCreation\_dateTimePicker.MaxDate = DateTime.Now;

toDateCreation\_dateTimePicker.MinDate = DateTime.Now.AddYears(-100);

toDateCreation\_dateTimePicker.MaxDate = DateTime.Now;

outDateExpiration\_dateTimePicker.MinDate = DateTime.Now.AddDays(1);

outDateExpiration\_dateTimePicker.MaxDate = DateTime.Now.AddYears(100);

toDateExpiration\_dateTimePicker.MaxDate = DateTime.Now.AddYears(100);

toDateExpiration\_dateTimePicker.MinDate = DateTime.Now.AddDays(1);

grocerie\_radioButton.Checked = true;

}

private void outPrice\_textBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Char.IsDigit(e.KeyChar) && !e.KeyChar.Equals(Convert.ToChar(Keys.Back)))

e.Handled = true;

else

e.Handled = false;

}

}

}

4.10. MainForm.cs

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Курсова

{

public partial class Main : Form

{

private Storage storage = new Storage();

private int index = 0;

public Main()

{

InitializeComponent();

}

private void openFileButton\_pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

openFileButton\_pictureBox.BorderStyle = BorderStyle.FixedSingle;

}

private void openFileButton\_pictureBox\_MouseLeave(object sender, EventArgs e)

{

openFileButton\_pictureBox.BorderStyle = BorderStyle.None;

}

private void addButton\_pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

addButton\_pictureBox.BorderStyle = BorderStyle.FixedSingle;

}

private void addButton\_pictureBox\_MouseLeave(object sender, EventArgs e)

{

addButton\_pictureBox.BorderStyle = BorderStyle.None;

}

private void addButton\_pictureBox\_Click(object sender, EventArgs e)

{

AddForm add = new AddForm();

add.ShowDialog();

bool check = false;

if (add.isGrocerie == 1)

{

foreach (Goods item in storage.GetStorage)

{

if (item is Groceries)

{

Groceries tmp = item as Groceries;

if (tmp == add.groceries)

{

item.quantity += add.groceries.quantity;

check = true;

foreach (ListViewItem i in items\_listView.Items)

{

if (i.Tag is Groceries)

{

Groceries grocerie = i.Tag as Groceries;

if (grocerie == add.groceries)

{

i.SubItems[4].Text = item.quantity.ToString();

break;

}

}

}

break;

}

}

}

if (check == false)

{

storage.AddItem(add.groceries);

if (add.groceries.isChanged == true)

MessageBox.Show("Price was reduced");

ListViewItem listViewItem = new ListViewItem(add.groceries.name, 0);

listViewItem.Tag = add.groceries;

listViewItem.SubItems.Add("Grocerie");

listViewItem.SubItems.Add(add.groceries.producer);

listViewItem.SubItems.Add(add.groceries.price.ToString());

listViewItem.SubItems.Add(add.groceries.quantity.ToString());

items\_listView.Items.Add(listViewItem);

}

}

else if (add.isGrocerie == 2)

{

foreach (Goods item in storage.GetStorage)

{

if (item is NonGroceries)

{

NonGroceries tmp = item as NonGroceries;

if (tmp == add.nonGroceries)

{

item.quantity += add.nonGroceries.quantity;

check = true;

foreach (ListViewItem i in items\_listView.Items)

{

if (i.Tag is NonGroceries)

{

NonGroceries nonGrocerie = i.Tag as NonGroceries;

if (nonGrocerie == add.nonGroceries)

{

i.SubItems[4].Text = item.quantity.ToString();

break;

}

}

}

break;

}

}

}

if (check == false)

{

storage.AddItem(add.nonGroceries);

if (storage.GetStorage[storage.GetStorage.Count - 1].isChanged)

MessageBox.Show("Price was increased");

ListViewItem listViewItem = new ListViewItem(add.nonGroceries.name, 1);

listViewItem.Tag = add.nonGroceries;

listViewItem.SubItems.Add("NonGrocerie");

listViewItem.SubItems.Add(add.nonGroceries.producer);

listViewItem.SubItems.Add(add.nonGroceries.price.ToString());

listViewItem.SubItems.Add(add.nonGroceries.quantity.ToString());

items\_listView.Items.Add(listViewItem);

}

}

}

private void saveFileButton\_pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

saveFileButton\_pictureBox.BorderStyle = BorderStyle.FixedSingle;

}

private void saveFileButton\_pictureBox\_MouseLeave(object sender, EventArgs e)

{

saveFileButton\_pictureBox.BorderStyle = BorderStyle.None;

}

private void saveFileButton\_pictureBox\_Click(object sender, EventArgs e)

{

if (storage.GetStorage.Count == 0)

{

MessageBox.Show("The storage is empty!", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

SaveFileDialog saveFile = new SaveFileDialog();

saveFile.Title = "Save file";

saveFile.Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

if (saveFile.ShowDialog() == DialogResult.OK && saveFile.FileName.Length > 0)

{

using (System.IO.StreamWriter sf = new System.IO.StreamWriter(saveFile.FileName))

{

foreach (Goods item in storage.GetStorage)

{

if (item is Groceries)

{

Groceries tmp = item as Groceries;

sf.WriteLine("Grocerie");

sf.WriteLine(tmp.name);

sf.WriteLine(tmp.producer);

sf.WriteLine(tmp.price);

sf.WriteLine(tmp.quantity);

sf.WriteLine(tmp.weight);

sf.WriteLine(tmp.dateCreation);

sf.WriteLine(tmp.dateExpiration);

}

else

{

NonGroceries tmp = item as NonGroceries;

sf.WriteLine("Non-Grocerie");

sf.WriteLine(tmp.name);

sf.WriteLine(tmp.producer);

sf.WriteLine(tmp.price);

sf.WriteLine(tmp.quantity);

sf.WriteLine(tmp.dimensions.height);

sf.WriteLine(tmp.dimensions.length);

sf.WriteLine(tmp.dimensions.width);

switch (tmp.group)

{

case DestinationGroup.Nothing:

sf.WriteLine(1);

break;

case DestinationGroup.Clothes:

sf.WriteLine(2);

break;

case DestinationGroup.Machinery:

sf.WriteLine(3);

break;

case DestinationGroup.Furniture:

sf.WriteLine(4);

break;

case DestinationGroup.Dishes:

sf.WriteLine(5);

break;

case DestinationGroup.Jewelry:

sf.WriteLine(6);

break;

}

}

}

sf.Close();

}

}

MessageBox.Show("File saved", "Saved", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private void openFileButton\_pictureBox\_Click(object sender, EventArgs e)

{

try

{

OpenFileDialog openFile = new OpenFileDialog();

openFile.Title = "Open from file";

openFile.Filter = "Text files (\*.txt)|\*.txt";

if (openFile.ShowDialog() == DialogResult.OK && openFile.FileName.Length > 0)

{

using (System.IO.StreamReader of = new System.IO.StreamReader(openFile.FileName))

{

while (!of.EndOfStream)

{

string type = of.ReadLine();

if (type == "Grocerie")

{

string name = of.ReadLine();

if (name == "")

name = "No\_name";

string producer = of.ReadLine();

if (producer == "")

producer = "No\_producer";

string strPrice = of.ReadLine();

int price;

if (int.TryParse(strPrice, out price) == false)

price = 0;

string strQuantity = of.ReadLine();

int quantity;

if (int.TryParse(strQuantity, out quantity) == false)

quantity = 0;

string strWeight = of.ReadLine();

int weight;

if (int.TryParse(strWeight, out weight) == false)

weight = 0;

string strDate = of.ReadLine();

DateTime dateCreation;

if (DateTime.TryParse(strDate, out dateCreation) == false)

dateCreation = DateTime.Now;

DateTime dateExpiration = Convert.ToDateTime(of.ReadLine());

strDate = of.ReadLine();

if (DateTime.TryParse(strDate, out dateExpiration) == false)

dateExpiration = DateTime.Now;

Groceries grocerie = new Groceries(name, producer, price, quantity, weight, dateCreation, dateExpiration);

foreach (Goods item in storage.GetStorage)

{

if (item is Groceries)

{

Groceries tmp = item as Groceries;

if (tmp == grocerie)

{

item.quantity += grocerie.quantity;

return;

}

}

}

storage.AddItem(grocerie);

ListViewItem listViewItem = new ListViewItem(grocerie.name, 0);

listViewItem.Tag = grocerie;

listViewItem.SubItems.Add("Grocerie");

listViewItem.SubItems.Add(grocerie.producer);

listViewItem.SubItems.Add(grocerie.price.ToString());

listViewItem.SubItems.Add(grocerie.quantity.ToString());

items\_listView.Items.Add(listViewItem);

}

else if (type == "Non-Grocerie")

{

string name = of.ReadLine();

if (name == "")

name = "No\_name";

string producer = of.ReadLine();

if (producer == "")

producer = "No\_producer";

string strPrice = of.ReadLine();

int price;

if (int.TryParse(strPrice, out price) == false)

price = 0;

string strQuantity = of.ReadLine();

int quantity;

if (int.TryParse(strQuantity, out quantity) == false)

quantity = 0;

Dimensions dimensions = new Dimensions();

string dimension = of.ReadLine();

if (int.TryParse(dimension, out dimensions.height) == false)

dimensions.height = 0;

dimension = of.ReadLine();

if (int.TryParse(dimension, out dimensions.length))

dimensions.length = 0;

dimension = of.ReadLine();

if (int.TryParse(dimension, out dimensions.width) == false)

dimensions.width = 0;

string strGroup = of.ReadLine();

int group;

if (int.TryParse(strGroup, out group) == false)

group = 0;

DestinationGroup destinationGroup = new DestinationGroup();

switch (group)

{

case 2:

destinationGroup = DestinationGroup.Clothes;

break;

case 3:

destinationGroup = DestinationGroup.Machinery;

break;

case 4:

destinationGroup = DestinationGroup.Furniture;

break;

case 5:

destinationGroup = DestinationGroup.Dishes;

break;

case 6:

destinationGroup = DestinationGroup.Jewelry;

break;

default:

destinationGroup = DestinationGroup.Nothing;

break;

}

NonGroceries nonGroceries = new NonGroceries(name, producer, price, quantity, dimensions, destinationGroup);

foreach (Goods item in storage.GetStorage)

{

if (item is NonGroceries)

{

NonGroceries tmp = item as NonGroceries;

if (tmp == nonGroceries)

{

item.quantity += nonGroceries.quantity;

return;

}

}

}

storage.AddItem(nonGroceries);

ListViewItem listViewItem = new ListViewItem(nonGroceries.name, 1);

listViewItem.Tag = nonGroceries;

listViewItem.SubItems.Add("NonGrocerie");

listViewItem.SubItems.Add(nonGroceries.producer);

listViewItem.SubItems.Add(nonGroceries.price.ToString());

listViewItem.SubItems.Add(nonGroceries.quantity.ToString());

items\_listView.Items.Add(listViewItem);

}

else

{

Exception exception = new Exception("File have mistakes");

throw exception;

}

}

}

}

}

catch (Exception exp)

{

MessageBox.Show(exp.Message);

}

}

private int Counter(string name)

{

int res = 0;

foreach (Goods item in storage.GetStorage)

{

if (item.name == name)

res += item.quantity;

}

return res;

}

private void items\_listView\_SelectedIndexChanged(object sender, EventArgs e)

{

if (items\_listView.SelectedIndices.Count == 0)

{

weightOrGroup\_label.Visible = false;

value1\_label.Visible = false;

dimensionsOrDateCreation\_label.Visible = false;

value2\_label.Visible = false;

dateExpiration\_label.Visible = false;

value3\_label.Visible = false;

piece\_label.Text = "0 pieces";

index = -1;

}

else

{

index = items\_listView.SelectedIndices.Count - 1;

var item = items\_listView.SelectedItems[items\_listView.SelectedIndices.Count - 1].Tag;

if (item is Groceries)

{

Groceries tmp = item as Groceries;

weightOrGroup\_label.Visible = true;

weightOrGroup\_label.Text = "Weight:";

value1\_label.Visible = true;

value1\_label.Text = tmp.weight.ToString();

dimensionsOrDateCreation\_label.Visible = true;

dimensionsOrDateCreation\_label.Text = "Creation date:";

value2\_label.Visible = true;

value2\_label.Text = tmp.dateCreation.ToShortDateString().ToString();

dateExpiration\_label.Visible = true;

value3\_label.Visible = true;

value3\_label.Text = tmp.dateExpiration.ToShortDateString().ToString();

piece\_label.Text = Counter(tmp.name).ToString() + " pieces";

}

else

{

value3\_label.Visible = false;

dateExpiration\_label.Visible = false;

NonGroceries tmp = item as NonGroceries;

weightOrGroup\_label.Visible = true;

weightOrGroup\_label.Text = "Group:";

value1\_label.Visible = true;

switch (tmp.group)

{

case DestinationGroup.Nothing:

value1\_label.Text = "Nothing";

break;

case DestinationGroup.Clothes:

value1\_label.Text = "Clothes";

break;

case DestinationGroup.Machinery:

value1\_label.Text = "Machinery";

break;

case DestinationGroup.Furniture:

value1\_label.Text = "Furniture";

break;

case DestinationGroup.Dishes:

value1\_label.Text = "Dishes";

break;

case DestinationGroup.Jewelry:

value1\_label.Text = "Jewelry";

break;

default:

break;

}

dimensionsOrDateCreation\_label.Visible = true;

dimensionsOrDateCreation\_label.Text = "Dimensions:";

value2\_label.Visible = true;

value2\_label.Text = tmp.dimensions.height.ToString() + "x" + tmp.dimensions.length.ToString() + "x" + tmp.dimensions.width.ToString();

piece\_label.Text = Counter(tmp.name).ToString() + " pieces";

}

}

}

private void deleteButton\_pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

deleteButton\_pictureBox.BorderStyle = BorderStyle.FixedSingle;

}

private void deleteButton\_pictureBox\_MouseLeave(object sender, EventArgs e)

{

deleteButton\_pictureBox.BorderStyle = BorderStyle.None;

}

private void deleteButton\_pictureBox\_Click(object sender, EventArgs e)

{

var items = items\_listView.SelectedItems;

foreach (var item in items)

{

ListViewItem listViewItem = item as ListViewItem;

storage.DeleteItem(listViewItem.Tag as Goods);

}

Update();

weightOrGroup\_label.Visible = false;

value1\_label.Visible = false;

dimensionsOrDateCreation\_label.Visible = false;

value2\_label.Visible = false;

dateExpiration\_label.Visible = false;

value3\_label.Visible = false;

}

private void reloadButton\_pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

reloadButton\_pictureBox.BorderStyle = BorderStyle.FixedSingle;

}

private void reloadButton\_pictureBox\_MouseLeave(object sender, EventArgs e)

{

reloadButton\_pictureBox.BorderStyle = BorderStyle.None;

}

private void reloadButton\_pictureBox\_Click(object sender, EventArgs e)

{

TimeSpan usingTime;

TimeSpan remainingTime;

double usingTimeI;

double remainingTimeI;

foreach (Goods item in storage.GetStorage)

{

if (item is Groceries)

{

Groceries tmp = item as Groceries;

if (tmp.dateExpiration < DateTime.Now)

{

storage.DeleteItem(item);

}

else if (item.isChanged == false)

{

usingTime = tmp.dateExpiration - tmp.dateCreation;

remainingTime = tmp.dateExpiration - DateTime.Now;

usingTimeI = usingTime.TotalSeconds;

remainingTimeI = remainingTime.TotalSeconds;

if ((usingTimeI \* 0.2) <= remainingTimeI)

{

item.price /= 2;

item.isChanged = true;

}

}

}

}

Update();

}

private void Update()

{

items\_listView.Items.Clear();

foreach (Goods item in storage.GetStorage)

{

ListViewItem listViewItem;

if (item is Groceries)

{

listViewItem = new ListViewItem(item.name, 0);

listViewItem.SubItems.Add("Grocerie");

}

else

{

listViewItem = new ListViewItem(item.name, 1);

listViewItem.SubItems.Add("NonGrocerie");

}

listViewItem.Tag = item;

listViewItem.SubItems.Add(item.producer);

listViewItem.SubItems.Add(item.price.ToString());

listViewItem.SubItems.Add(item.quantity.ToString());

items\_listView.Items.Add(listViewItem);

}

}

private void items\_listView\_ColumnClick(object sender, ColumnClickEventArgs e)

{

switch (items\_listView.Sorting)

{

case SortOrder.None:

items\_listView.Sorting = SortOrder.Ascending;

break;

case SortOrder.Ascending:

items\_listView.Sorting = SortOrder.Descending;

break;

case SortOrder.Descending:

items\_listView.Sorting = SortOrder.Ascending;

break;

}

items\_listView.ListViewItemSorter = new CompareItems(e.Column, items\_listView.Sorting);

}

private void filterButton\_pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

filterButton\_pictureBox.BorderStyle = BorderStyle.FixedSingle;

}

private void filterButton\_pictureBox\_MouseLeave(object sender, EventArgs e)

{

filterButton\_pictureBox.BorderStyle = BorderStyle.None;

}

private void filterButton\_pictureBox\_Click(object sender, EventArgs e)

{

FilterForm filterForm = new FilterForm(storage);

filterForm.ShowDialog();

}

private void editPriceButton\_pictureBox\_MouseLeave(object sender, EventArgs e)

{

editPriceButton\_pictureBox.BorderStyle = BorderStyle.None;

}

private void editPriceButton\_pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

editPriceButton\_pictureBox.BorderStyle = BorderStyle.FixedSingle;

}

private void editPriceButton\_pictureBox\_Click(object sender, EventArgs e)

{

if (items\_listView.SelectedIndices.Count != 0)

{

label.Visible = true;

newPrice\_textBox.Visible = true;

acceptButton\_pictureBox.Visible = true;

newPrice\_textBox.Focus();

}

}

private void acceptButton\_pictureBox\_MouseMove(object sender, MouseEventArgs e)

{

acceptButton\_pictureBox.BackColor = Color.AliceBlue;

}

private void acceptButton\_pictureBox\_MouseLeave(object sender, EventArgs e)

{

acceptButton\_pictureBox.BackColor = SystemColors.ActiveCaption;

}

private void acceptButton\_pictureBox\_Click(object sender, EventArgs e)

{

if (newPrice\_textBox.Text != String.Empty)

{

storage.GetStorage[index].price = int.Parse(newPrice\_textBox.Text);

Update();

}

label.Visible = false;

newPrice\_textBox.Text = "";

newPrice\_textBox.Visible = false;

acceptButton\_pictureBox.Visible = false;

}

private void newPrice\_textBox\_Leave(object sender, EventArgs e)

{

label.Visible = false;

newPrice\_textBox.Text = "";

newPrice\_textBox.Visible = false;

acceptButton\_pictureBox.Visible = false;

}

private void newPrice\_textBox\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Char.IsDigit(e.KeyChar) && !e.KeyChar.Equals(Convert.ToChar(Keys.Back)))

e.Handled = true;

else

e.Handled = false;

}

}

}

4.11. Program.cs

using System;

using System.Windows.Forms;

namespace Курсова

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Main());

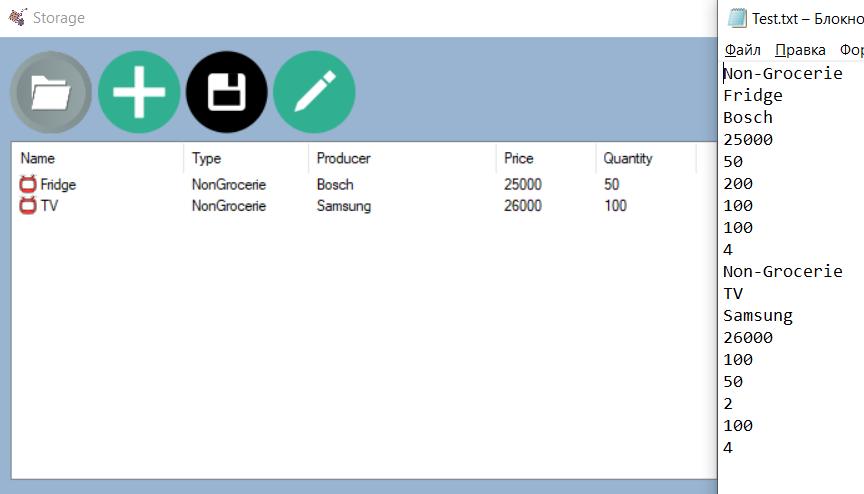
}

}

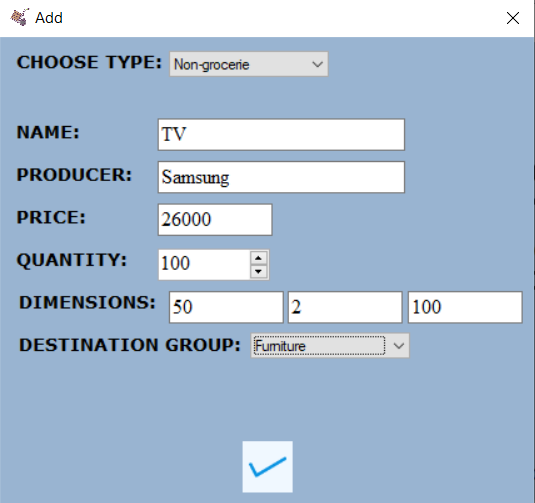
}

**5.** **Протокол роботи програми**

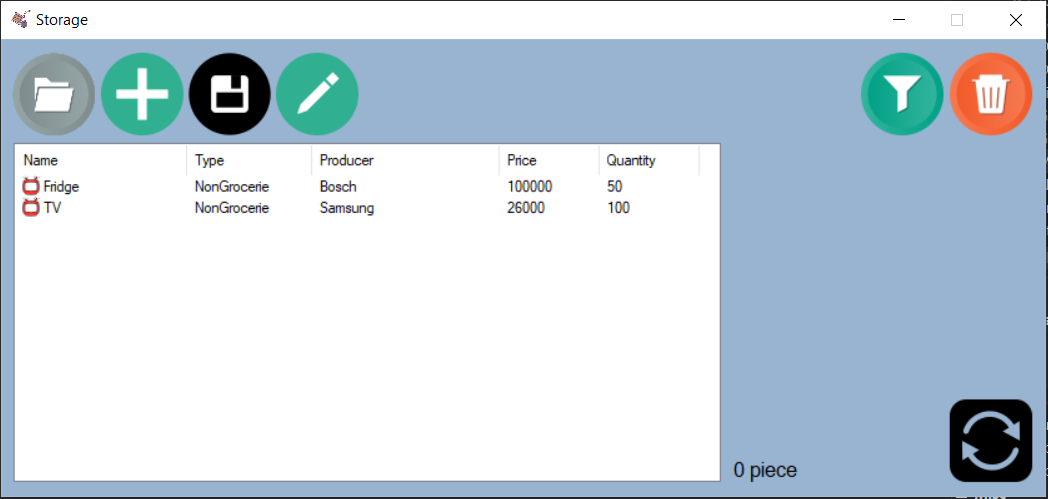
**Додавання товарів за допомогою файлу**

****

**Додавання товарів вручну**

****

**Список товарів**

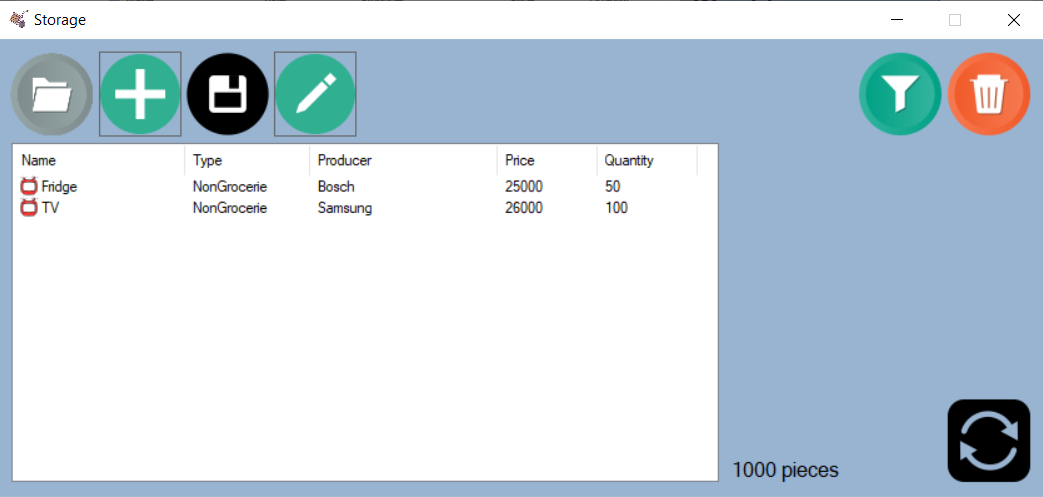
****

**Вивід повної інформації про товар**

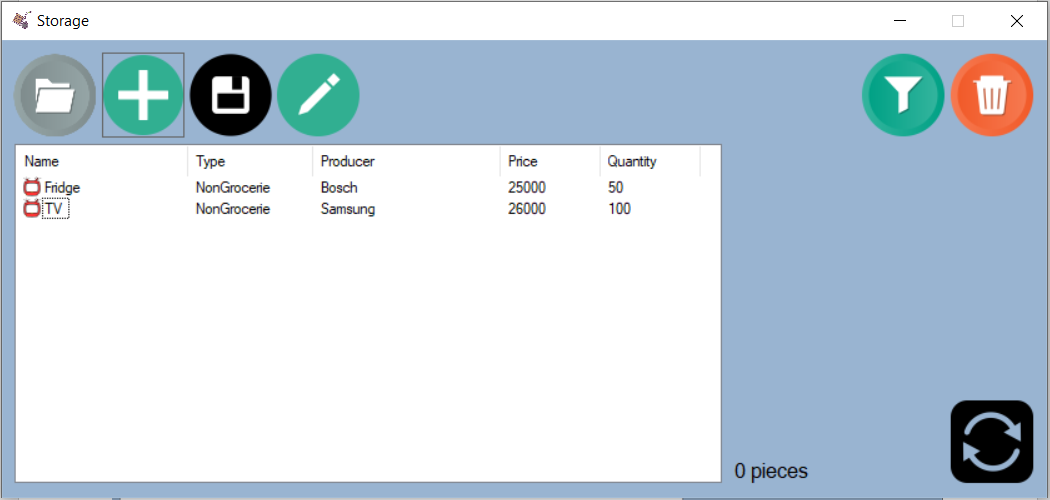
****

**Видалення товару зі списку**

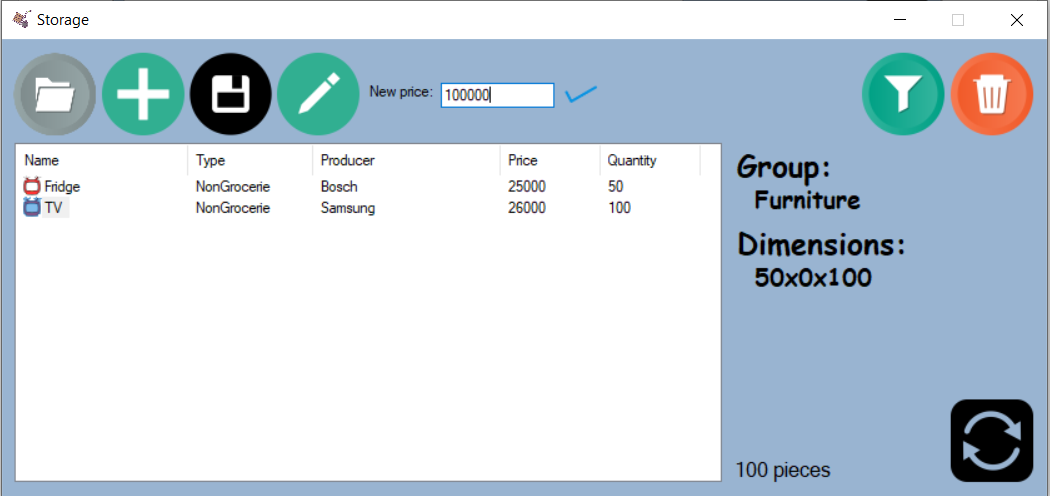
****

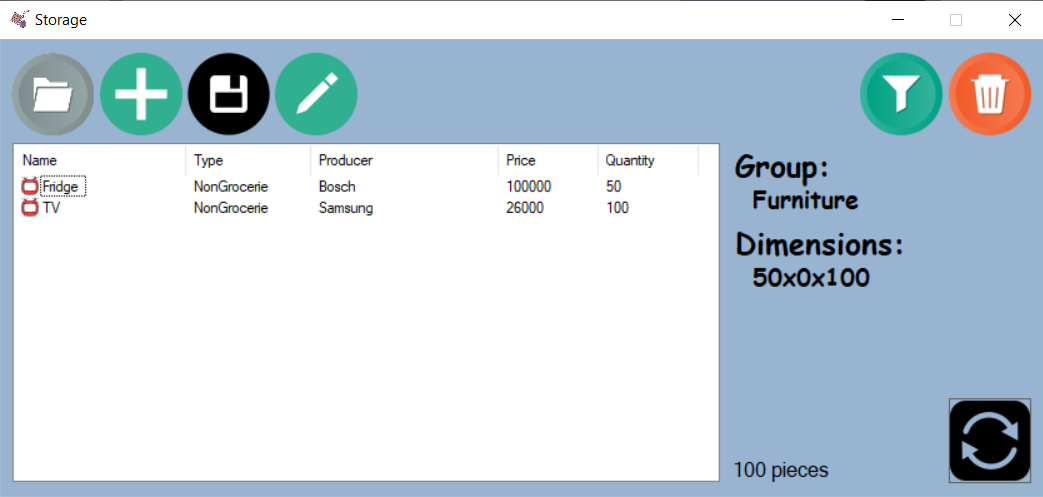
****

**Сортування списку(в даному випадку ім’я в алфавітному порядку)**

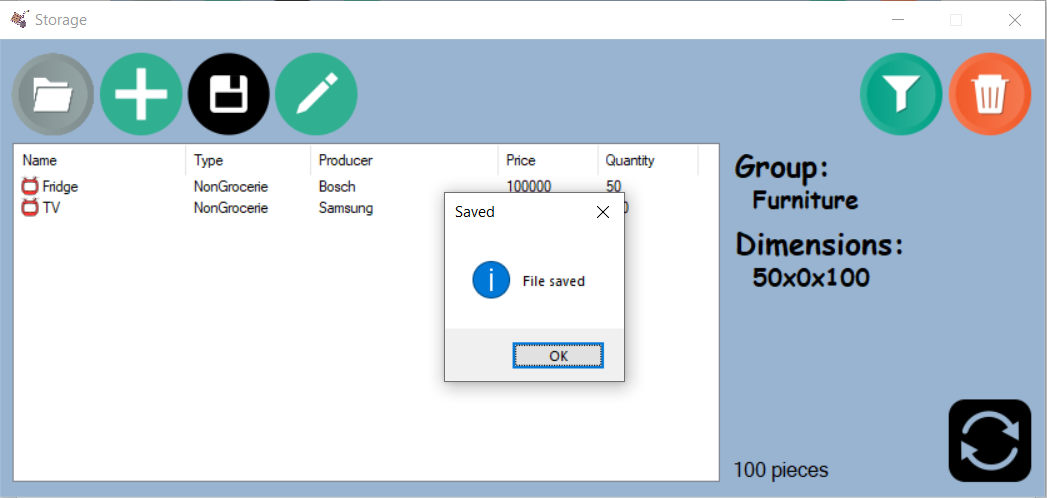
****

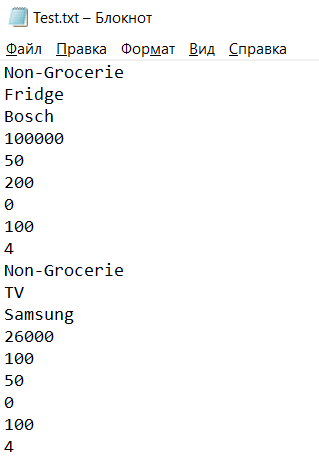
**Зміна ціни**

****

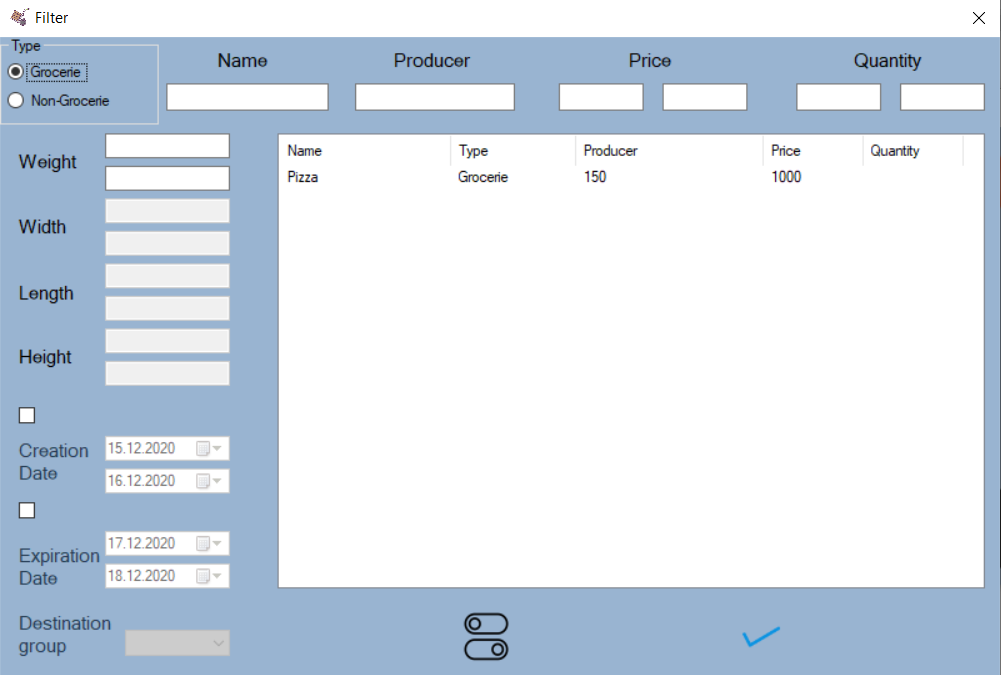
****

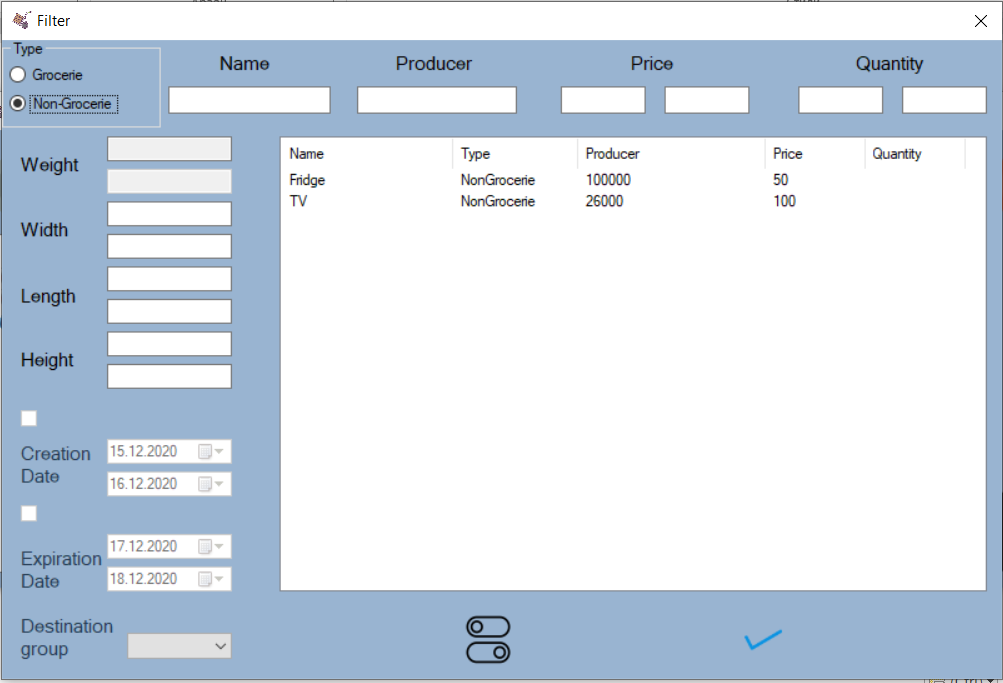
**Збереження в файл**

****

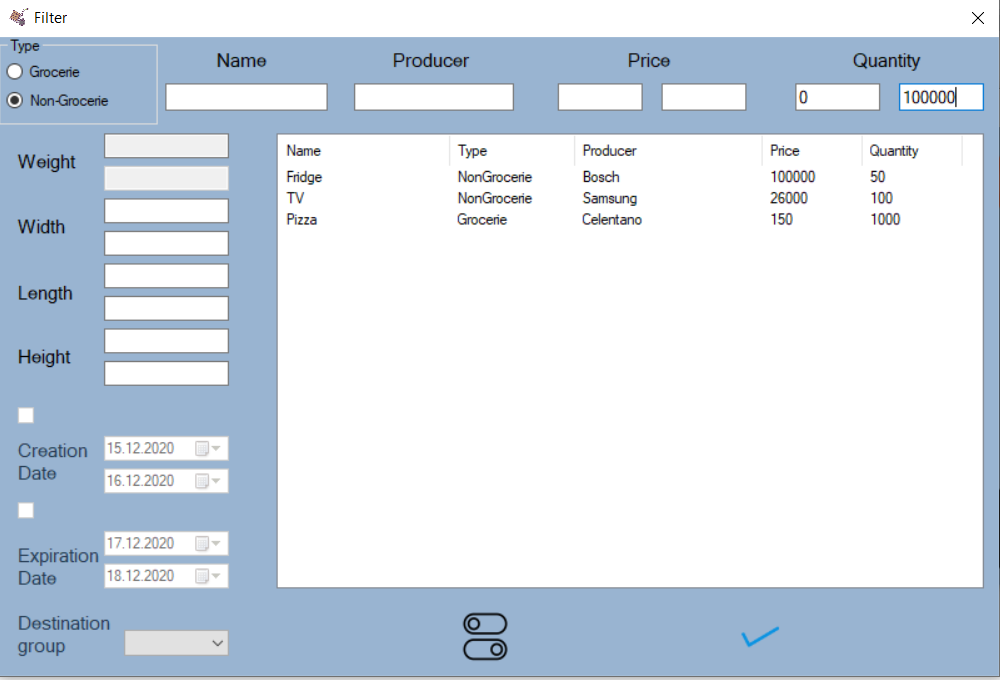
****

**Розподіл товарів по групах**

****

****

**Фільтрація товарів (в даному випадку діапазон ціни)**

****

**Підрахунок сумарної кількості товарів обраної назви**

****

**6.** **Інструкція користувача**

*1. Компоненти ПЗ*

Пакет розроблено мовою програмування C#.NET Framework 4.8 у середовищі розробки MS Visual Studio і може експлуатуватися під управлінням сімейства операційних систем Windows. Під час проектування підсистем відбувся об’єктно-орієнтований підхід. Усі класи документувались і семантично.

Для коректної роботи пакету необхідна користувацька машина з процесором з тактовою частотою не меншою за 1.6 Hz та оперативною пам’яттю не меншою за 1 Gb. Перелік необхідних файлів для коректного запуску програми наведено у наведеній нижче таблиці базових функцій.

*2. Встановлення ПЗ*

Для початку роботи з даним ПЗ вам необхідно розархівувати файли з архіву “Курсова” та відкрити виконавчий файл “Курсова.exe”.

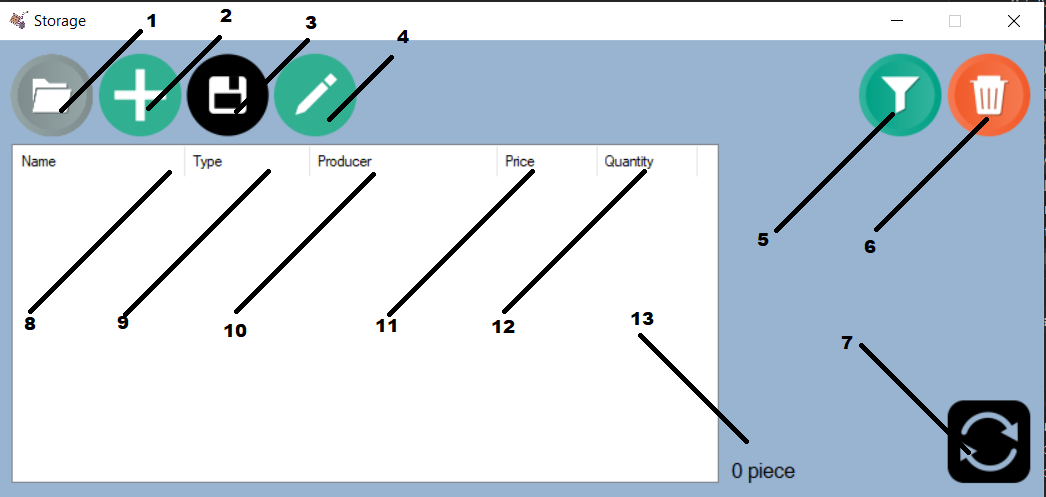
У встановленні додаткового ПЗ для роботи програми необхідності немає.

*3. Налаштування ПЗ*

Для коректної роботи програми не потрібно додатково нічого налаштовувати.

*4. Базові функції ПЗ:*

*Графічний інтерфейс користувача:*



Головний екран користувача

1. Кнопка для зчитування товарів з файлу;

2. Кнопка для додавання товарів вручну;

3. Кнопка для зберігання товарів у файл;

4. Кнопка для зміни вартості виділеного товару:

5. Кнопка для відкриття вікна, де можна відфільтрувати товари, або знайти унікальні;

6. Кнопка для видалення виділеного товару;

7. Кнопка для оновлення дати в програмі;

8. Поле назви товару;

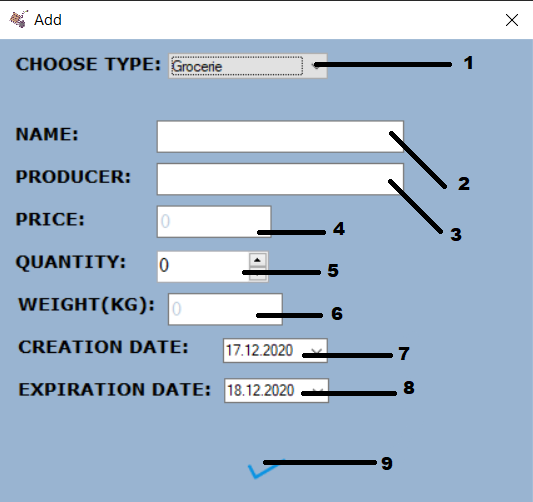
9. Поле типу товару;

10. Поле виробника товару;

11. Поле ціни товару;

12. Поле кількості товару;

13. Поле, де після виділення одного товару буде писати сумарну кількість товарів з таким самим ім’ям.



Вікно додавання товару вручну, якщо користувач бажає додати продовольчі товари

1. Поле, де можна обрати тип товару (в нашому випадку - продовольчий);

2. Поле вводу назви товару;

3. Поле вводу виробника товару;

4. Поле вводу ціни товару (максимум 2147483647);

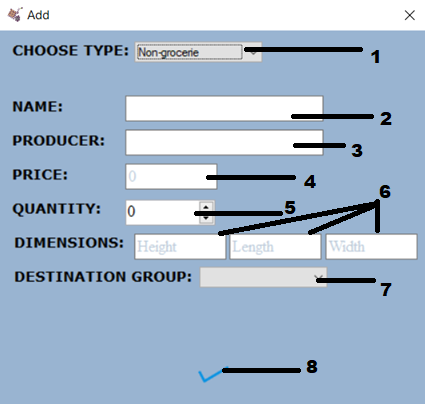
5. Поле вводу кількості товару (максимум 1000);

6. Поле вводу ваги товару (максимум 2147483647);

7. Поле вибору дати створення товару;

8. Поле вибору терміну придатності товару;

9. Кнопка підтведження, яка закриває вікно, додає товар на склад та виводить його в список.



Вікно додавання товару вручну, якщо користувач бажає додати непродовольчі товари

1. Поле, де можна обрати тип товару (в нашому випадку - непродовольчий);

2. Поле назви товару;

3. Поле виробника товару;

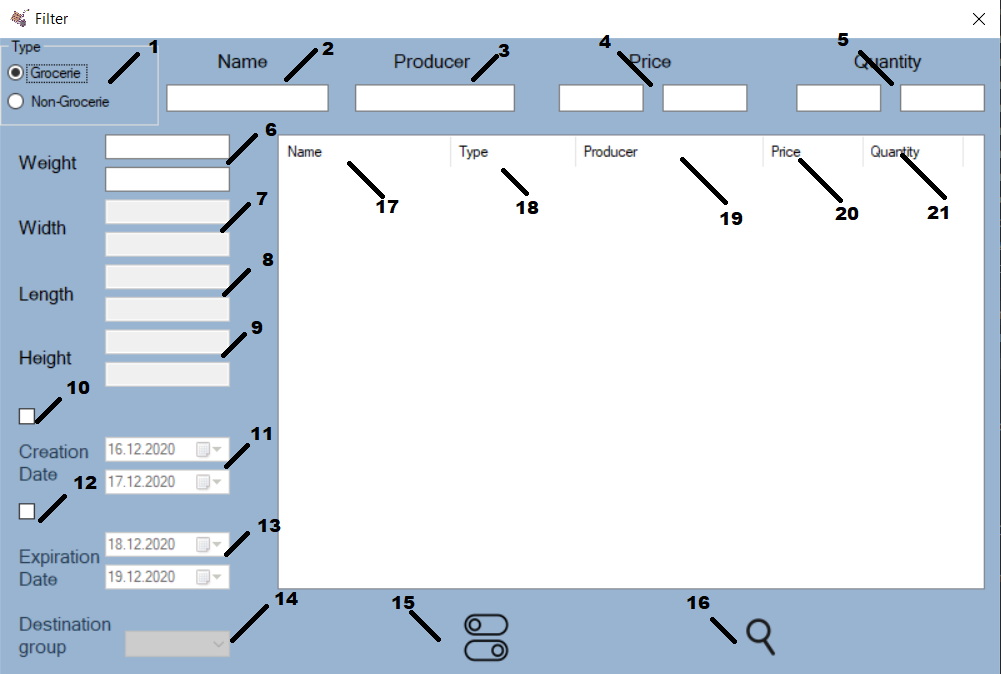
4. Поле вводу ціни товару (максимум 2147483647);

5. Поле вводу кількості товару (максимум 1000);

6. Поля габаритів товарів (висота, довжина і ширина максимум по 2147483647);

7. Поле де можна обрати групу призначення товару (одяг, прикраси і тд);

8. Кнопка підтведження, яка закриває вікно, додає товар на склад та виводить його в список.



Вікно, де користувач може відфільтрувати товари на складі

1. Місце вибору користувачем, який тип товару він хоче побачити;

2. Поле вводу назви товару;

3. Поле вводу виробника товару;

4. Поля вводу діапазону ціни товару;

5. Поля вводу діапазону кількості товару;

6. Поля вводу діапазону ваги товару (працює лише тоді, коли користувач обрав пункт Grocerie);

7. Поля вводу діапазону ширини товару (працює лише тоді, коли користувач обрав пункт Non-Grocerie);

8. Поля вводу діапазону довжини товару (працює лише тоді, коли користувач обрав пункт Non-Grocerie);

9. Поля вводу діапазону висоти товару (працює лише тоді, коли користувач обрав пункт Non-Grocerie);

10. Кнопка вибору користувача, чи хоче він фільтрувати також за діапазоном дати створення (якщо галочка стоїть, то активовуються наступні два поля) (працює лише тоді, коли користувач обрав пункт Grocerie);

11. Поля вибору діапазону дати створення товару (працює лише тоді, коли користувач сам підтвердив те, що він хоче по цьому також фільтрувати);

12. Кнопка вибору користувача, чи хоче він фільтрувати також за діапазоном терміну придатності (якщо галочка стоїть, то активовуються наступні два поля) (працює лише тоді, коли користувач обрав пункт Grocerie);

13. Поля вибору діапазону терміну придатності товару (працює лише тоді, коли користувач сам підтвердив те, що він хоче по цьому також фільтрувати);

14. Поле вибору групи призначення товару (працює лише тоді, коли користувач обрав пункт Non-Grocerie);

15. Кнопка пошуку унікального товару (працює лише в тому випадку, якщо всі поля заповнені, відносно типу товару);

16. Кнопка фільтрації товару відносно зазначених користувачем критеріїв;

17. Поле назви знайденого товару:

18. Поле типу знайденого товару;

19. Поле виробника знайденого товару;

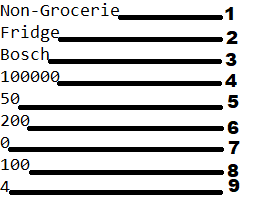
20. Поле ціни знайденого товару;

21. Поле кількості знайденого товару.

*Таблиця базових функцій:*

|  |  |  |
| --- | --- | --- |
| Назва | Вигляд кнопки | Функція |
| Відкрити з файлу |  | Відкриття та зчитування товарів з файлу |
| Збереження в файл |  | Збереження складу товарів в текстовий файл |
| Додавання товару вручну |  | Кнопка для переходу в вікно додавання товару вручну |
| Зміна ціни товару |  | Кнопка для відкриття меню зміни ціни товару |
| Видалення товару зі складу |  | Кнопка для видалення виділеного елемента/ів |
| Оновлення часу в програмі |  | Кнопка для оновлення часу в програмі |
| Відкриття меню фільтрування |  | Кнопка для відкривання меню фільтрації |
| Фільтрація товарів |  | Кнопка пошуку товарів згідно вказаних критеріїв |
| Пошук унікальних товарів |  | Кнопка для пошуку товарів, які не співпадають ні з одним критерієм пошуку |
| Підтвердження дії |  | Кнопка, яка підтвердує якусь дію |

**7.** **Структура файлу вхідних даних**



Структура файлу вводу даних, якщо користувач вводить непродовольчий товар

1. Тип товару;

2. Назва товару;

3. Виробник товару;

4. Вартість товару;

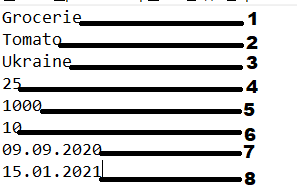
5. Кількість товару;

6. Висота товару;

7. Довжина товару;

8. Ширина товару;

9. Група призначення товару.



Структура файлу вводу даних, якщо користувач вводить продовольчий товар

1. Тип товару;

2. Назва товару;

3. Виробник товару;

4. Ціна товару;

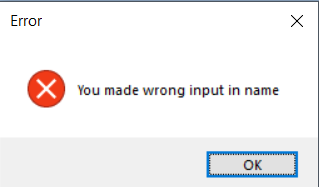
5. Кількість товару;

6. Вага товару;

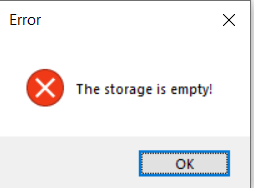
7. Дата виготовлення товару;

8. Термін придатності товару.

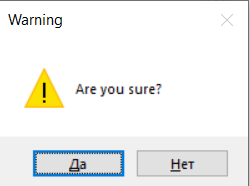
**8.** **Опис виняткових ситуацій**

****

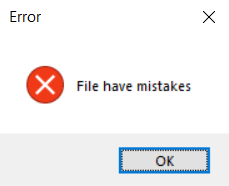
Попередження-помилка щодо неправильного вводу даних



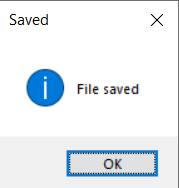
Попередження-помилка щодо того, що при збереженні пустого сховища файл нічого зберігати



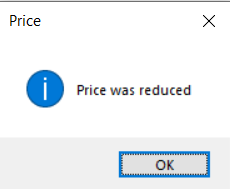
Повідомлення-підтвердження про видалення товарів з складу



Повідомлення-помилка про неправильність запису в відкритому файлі



Інформаційне повідомлення, що файл збережено успішно



Інформаційне повідомлення, що ціну було зменшено, через термін придатності, або збільшено через перевищення норм габаритів

**Висновки**

Виконуючи дану курсову роботу, я створив програму, що моделює роботу складу продовольчих та непродовольчих товарів. Для реалізації програми я використав середовище розробки MS Visual Studio та мову програмування C#.NET. Програма може зберігати, видаляти, сортувати, фільтрувати та змінювати дані, які вносить користувач. Також я розробив власний, інтуїтивно зрозумілий інтерфейс, що спрощує зв’язок “користувач-програма”. Для створення програми, я використав свої знання зі сфери об’єктно-орієнтованого програмування, а також вступу до інженерії програмного забезпечення.

**Список використаних джерел**

1. Документация по C# [Електронний ресурс] – Режим доступу: <https://docs.microsoft.com/ru-ru/dotnet/csharp/>

2. Язык C# и платформа .NET Core [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/tutorial/1.1.php>

3. Stack overflow [Електронний ресурс] – Режим доступу: https://stackoverflow.com/questions/7838079/why-cant-we-assign-a-foreach-iteration-variable-whereas-we-can-completely-modi