Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

КУРСОВА РОБОТА ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліні “ Об’єктно - орієнтоване програмування”

ЛОМБАРД

Керівник , проф. Бондарєв В.М.

Студент гр. ПЗПІ-22-1 Токар Д. Ю.

Комісія:

Проф. Бондарєв В.М.

Ст. викл. Черепанова Ю.Ю

Ст. викл. Ляпота В.М.

Харків 2023

# РЕФЕРАТ

Пояснювальна записка до курсової роботи: 44 с., 27 рис., 1 додаток.

ЛОТ, ЛОМБАРД, ЗВІТ, ООП, .NET, МОВА C#

Метою роботи є розробка програми «Ломбард», який буде надавати користувачу інформацію про лоти у ломбарді.

В результаті отримана програма, що дозволяє зберігати список лотів ломбарду, характеристики кожного лоту, такі як: назва, власник, оціночна вартість, видана сума, дата отримання та термін зберігання. Є можливість утворювати нові списки лотів, додавати, видаляти та редагувати лоти.

В процесі розробки використано середовища Microsoft Visual Studio 2022, фреймворк Windows Forms, платформи .NET 6.0, мова програмування C#

**ЗМІСТ**

Вступ ……………………………………………………………………………...3

1 Опис вимог………………………………………………………………….......4

2 Проектування програми………………………………………………………11

3 Інструкція користувача………………………………………………………..13

Висновки………………………………………………………………………....25

Додаток А………………………………………………………………………...26

**Вступ**

Тема

Ломбард. База збережених товарів і нерухомості: анкетні дані клієнта, найменування товару, оціночна вартість; сума, видана під заставу, дата здачі, термін зберігання. Операції прийому товару, повернення, продажу після закінчення терміну зберігання.

Мета

Метою створення програми є полегшення типової діяльності працівника ломбарду.

**1 ОПИС ВИМОГ**

* 1. Сценарії використання

**Сценарій 1. Запуск програми**

*Основний сценарій*

Запустивши програму користувач бачить повний список лотів ломбарду.

**Сценарій 2. Додавання лоту**

*Передумова*

За допомогою випадаючого меню “Lot” користувач обирає “Add”.

*Основний сценарій*

1. Програма відкриває вікно додавання лоту.
2. Користувач заповнює відповідні дані і натискає кнопку “Save”.
3. Програма перевіряє коректність даних в полях, створює новий лот і додає його до списку лотів.

*Додатковий сценарій*

1. Програма відкриває вікно додавання книги.
2. Користувач заповнює поля у вікні додавання і натискає кнопку “Save”.
3. Програма встановлює, що дані не коректні.
4. Програма сповіщає користувача про помилку в даних і про те, яким чином її виправити.

**Сценарій 3. Редагування лоту**

*Передумова*

Користувач натискає на потрібний лот і за допомогою випадаючого меню “Lot” користувач обирає “Edit”.

*Основний сценарій*

1. Програма відкриває вікно редагування лоту.
2. Користувач заповнює відповідні дані і натискає кнопку “Save”.
3. Програма перевіряє коректність даних в полях і редагує лот.

*Додатковий сценарій*

1. Програма відкриває вікно редагування книги.
2. Користувач заповнює поля у вікні редагування і натискає кнопку “Save”.
3. Програма встановлює, що дані не коректні.
4. Програма сповіщає користувача про помилку в даних і про те, яким чином її виправити.

**Сценарій 4. Видалення лоту**

*Передумова*

Користувач натискає на конкретний лот і за допомогою випадаючого меню “Lot” користувач обирає “Delete”.

*Основний сценарій*

1. Користувач отримає питання у діалоговому вікні «Do you want to delete this lot?» та натискає «Yes».
2. Програма видаляє вибраний лот.

*Додатковий сценарій*

1. Користувач отримає питання у діалоговому вікні «Do you want to delete this lot?» та натискає «No».
2. Програма скасовує видалення та повертає користувача на сторінку з лотами.

**Сценарій 5.** **Продаж лоту після закінчення терміну зберігання.**

*Передумова*

Термін зберігання лоту скінчився і за допомогою випадаючого меню “Lot” користувач обирає “Sell”.

*Основний сценарій*

1. Користувач отримає питання у діалоговому вікні «Do you want to sell this lot?» та натискає «Yes».
2. Програма видаляє вибраний лот.

*Додатковий сценарій*

1. Користувач отримає питання у діалоговому вікні «Do you want to sell this lot?» та натискає «No».
2. Програма скасовує продаж та повертає користувача на сторінку з лотами.

**Сценарій 6. Пошук лоту**

*Передумова*

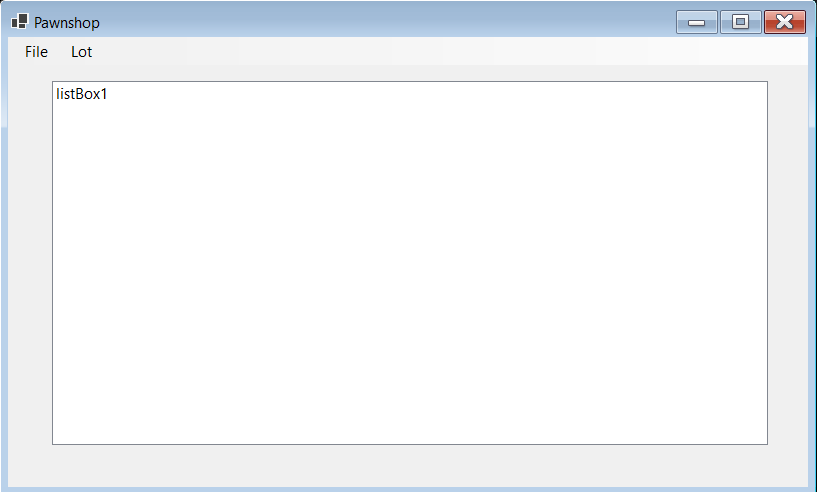
За допомогою випадаючого меню “Lot” користувач обирає “Search”.

*Основний сценарій*

1. Програма відкриває вікно пошуку лоту.
2. Користувач вводить дані у поле і натискає кнопку “Search”.
3. Програма виконує пошук по всім параметрам за введеними даними.
4. Програма відображає користувачу список лотів що підходять за введеними даними.
   1. Функціональні вимоги

**Вимога 1. Головне меню**

Після запуску програми користувач бачить перед собою список всіх лотів та спливаюче меню керування файлами та додаванням лотів у верхньому лівому куті (рис. 1.1).

  
Рисунок 1.1 – Головне меню

**Вимога 2.** **Додавання лотів**

Після натискання користувачем пункту “Add” в випадаючому меню відкривається вікно додання лотів, де користувач може вписувати дані у відповідні поля (рис. 1.2).

Текстовими полями є назва предмету та повне ім’я власника.

Числовими є ціна, видана сума і кількість днів до кінця терміну збереження.  
У числові поля користувач може вводити лише додатні числа.

Користувач повинен заповнити всі поля.

Номер (id) для лоту програма встановлює автоматично.

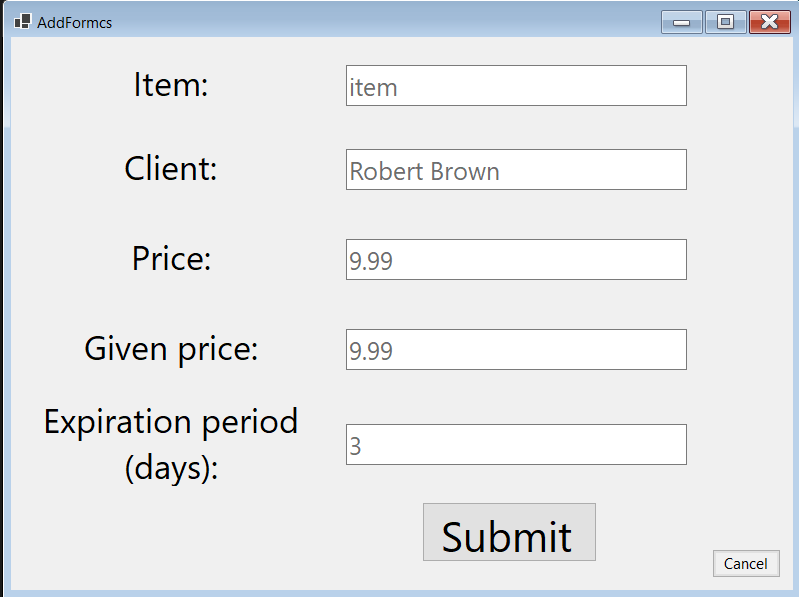


Рисунок 1.2 – Меню додавання лотів

**Вимога 3.** **Редагування лотів**

Після натискання користувачем пункту “Edit” при натисненні на певний лот відкривається вікно редагування лотів, де користувач може редагувати дані у відповідних полях (рис. 1.3).

Не всі відображенні дані доступні для редагування, доступні поля: назва предмету, повне ім’я власника, ціна, видана сума, термін збереження.

Текстовими полями є назва предмету та повне ім’я власника.

Числовими є ціна, видана сума і кількість днів до кінця терміну збереження.  
У числові поля користувач може вводити лише додатні числа.

Користувач може заповнити певні поля, а ті що не були заповнені не змінять свого минулого значення.

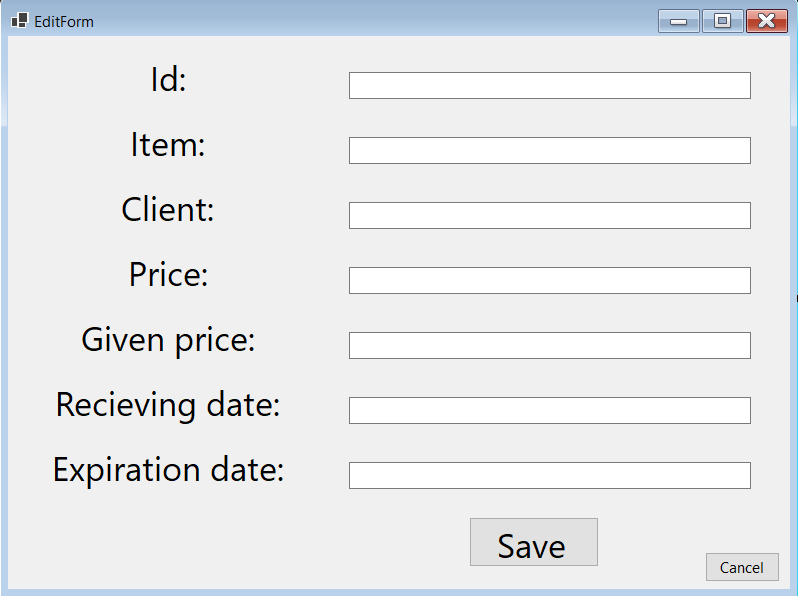


Рисунок 1.3 – Меню редагування лотів

**Вимога 4.** **Видалення лотів**

Після натискання користувачем пункту “Delete” при натисненні на певний лот відкривається вікно підтвердження видалення лоту, де користувач підтверджує видалення обраного лоту (рис. 1.4).

При натисканні «Yes» програма видаляє обраний лот.

При натисканні «No» програма повертає користувача на головний екран.

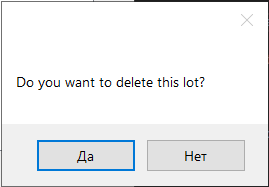


Рисунок 1.4 – Підтвердження видалення лоту

**Вимога 5.** **Продаж лотів**

Після натискання користувачем пункту “Sell” при натисненні на певний лот у якого скінчився термін зберігання відкривається вікно підтвердження продажу лоту, де користувач підтверджує продаж обраного лоту (рис. 1.5).

При натисканні «Yes» програма видаляє обраний лот.

При натисканні «No» програма повертає користувача на головний екран.

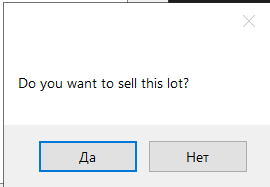


Рисунок 1.5 – Підтвердження продажу лоту

**Вимога 6.** **Пошук лотів**

Після натискання користувачем пункту “Search” в випадаючому меню відкривається вікно пошуку лотів, де користувач може вписати дані у поле пошуку (рис. 1.6).

При натисканні кнопки «Search» програма виконує пошук серед лотів по всім параметрам лоту і виводить результат на екран користувача.

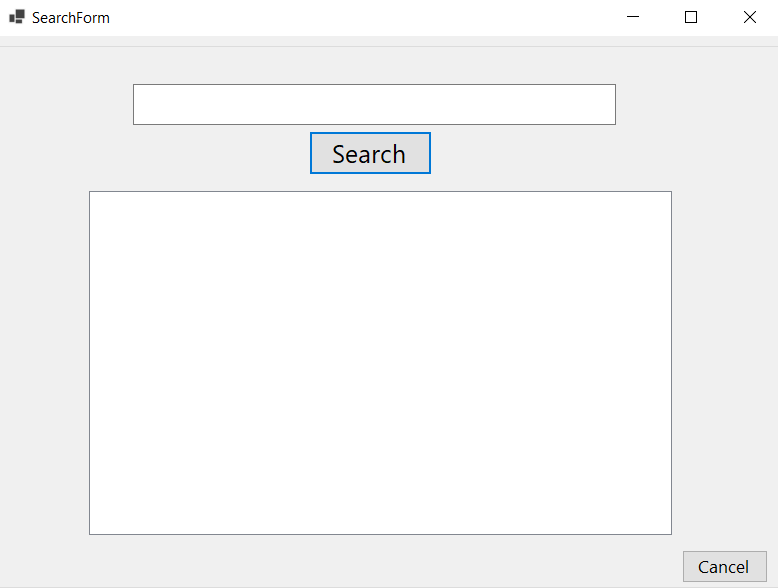


Рисунок 1.6 – Меню пошуку лотів

**2 ПРОЕКТУВАННЯ ПРОГРАМИ**

2.1 Структура програми

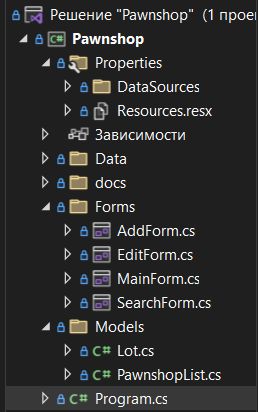
Програма складається з наступних основних частин:

* *Головне вікно*. Це вхідна точка програми, де користувач може переглядати список всіх лотів, переходити до інших форм програми.
* *Вікно додавання лоту*. В цьому вікні користувач може додавати нові лоти, вводячи назву предмета, власника, оціночну вартість, видану суму та термін зберігання.
* *Вікно редагування лоту*. В цьому вікні користувач може редагувати вже існуючі лоти, змінюючи їх назву предмета, власника, оціночну вартість, видану суму, дату отримання та термін зберігання.
* *Вікно пошуку лотів*. В цьому вікні користувач може здійснювати пошук за будь-яким параметром лотів, вводячи його у відповідне поле.

2.2 Код програми

Програма була розроблена на мові програмування C# з використанням принципів об’єктно орієнтованого програмування (ООП). Класи програми розподілені між собою за категоріями та структуровані у файлах(рис. 2.1). Основні класи програми включають:

* Клас Lot. Цей клас представляє окремий лот з полями для: номеру (id), назви предмету, імені власника, оціночної вартості, виданої суми, дати отримання, терміну зберігання.
* Клас PawnshopList. Цей клас представляє увесь ломбард, у ньому присутні: змінна для визначення унікального номеру (id), булева змінна для визначення стану ломбарду, список з лотів.
* Класи форм. Це класи, що відображають окремі вікна програми (MainForm, AddForm, EditForm, SearchForm), з методами для обробки дій користувача та взаємодії з об’єктами зі списку PawnshopList.

  
Рисунок 2.1 – Структура файлів проекту

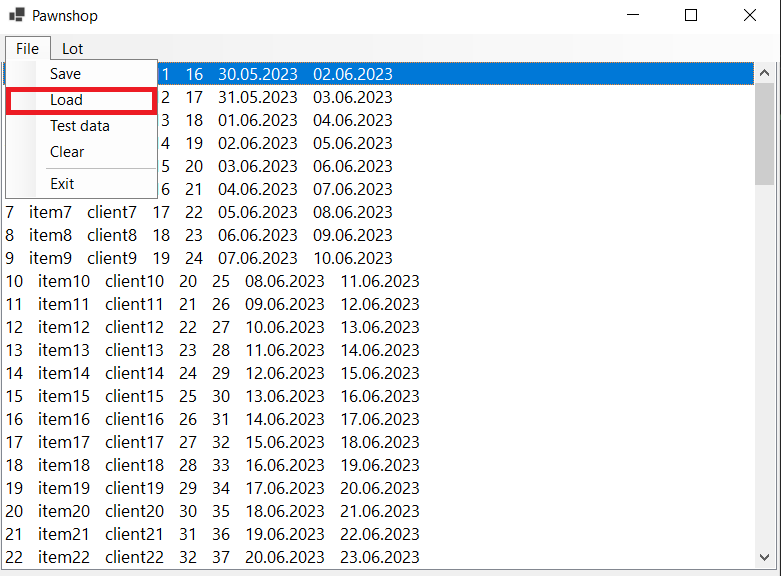
**3 Інструкція користувача**

**3.1 Завантаження даних**

Користувач натискає кнопку “Load” (рис. 3.1) у випадаючому меню у вкладці “File”.

У результаті програма завантажує збережені раніше дані з файлу.

Рисунок 3.1 – Кнопка “Load”

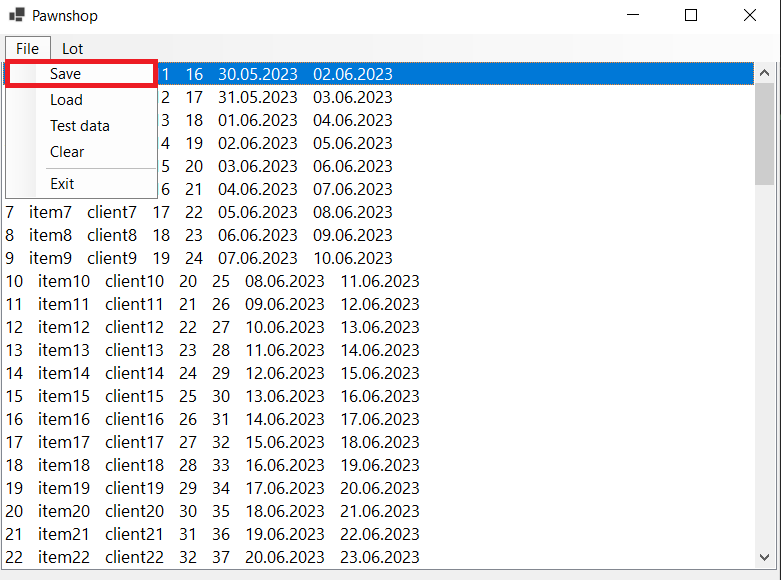


**3.2 Збереження даних**

Користувач натискає кнопку “Save” (рис. 3.2) у випадаючому меню у вкладці “File.

У результаті програма зберігає поточну інформацію у файлі.

Рисунок 4.2 – Кнопка “Save”

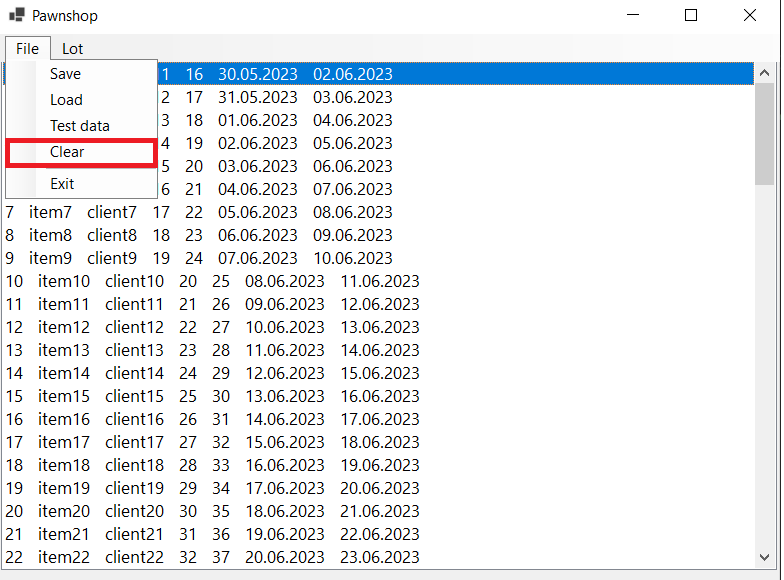


**3.3 Очищення поточних даних**

Користувач натискає кнопку “Clear” (рис. 3.3) у випадаючому меню у вкладці “File”.

У результаті програма видаляє всю поточну інформацію що відображається на екрані (не впливає на інформації збережену у файлі).

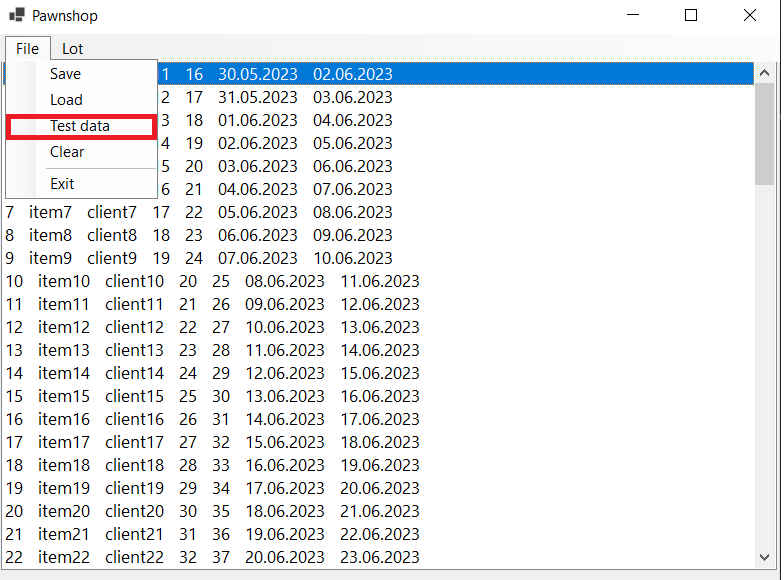
Рисунок 3.3 – Кнопка “Clear”



**4.4 Створення тестових даних**

Користувач при натискає кнопку “Test data” (рис. 3.4) у випадаючому меню у вкладці “File”.

У результаті програма додає до існуючого списку 10 тестових лотів.

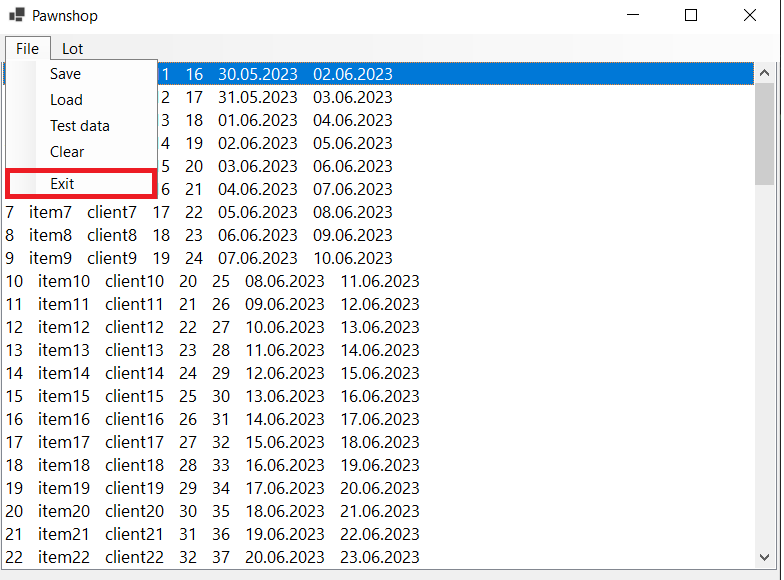
  
Рисунок 3.4 – Кнопка “Test data”

**3.5 Вихід з програми**

Користувач натискає кнопку “Exit” (рис. 3.5) у випадаючому меню у вкладці “File”.

У результаті якщо вміст списку не змінився програма закривається без додаткових операцій, але якщо список відрізняється програма запитує у користувача чи треба зберігати поточний список у файл, якщо так то програми виконує збереження списку і повне закриття.

Рисунок 3.5 – Кнопка “Exit”



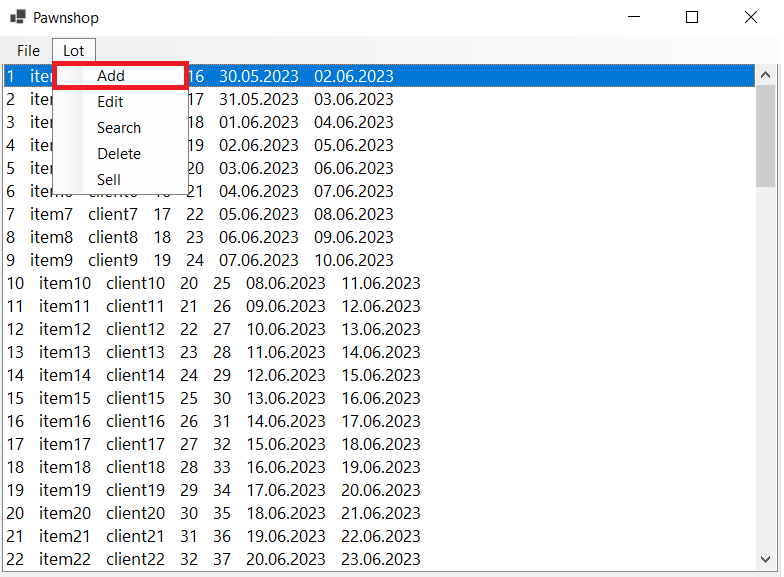
**3.6 Додавання лоту**

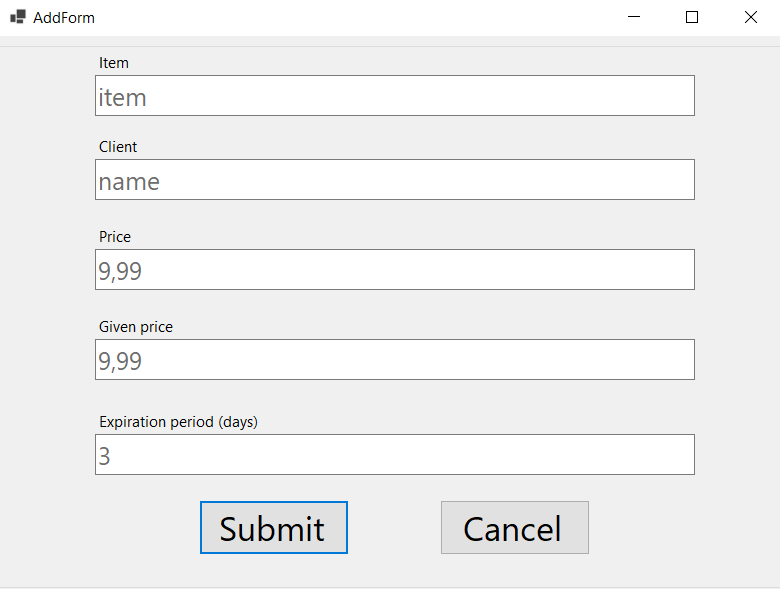
Послідовність дій користувача:

1. Користувач натискає кнопку “Add” (рис. 3.6) у спливаючому меню у вкладці “Lot”
2. У формі для додавання нових лотів (рис. 3.7) користувач вводить дані лоту: назву предмета, власника, оціночну вартість, ціну видану власнику та термін зберігання.
3. Користувач натискає кнопку “Submit”.

У результаті Id заповнюється програмою автоматично та є унікальним для кожного лоту. Дата отримання встановлюється відповідно до часу додавання елементу. Програма створює новий лот за отриманими від користувача даними та додає його до поточного списку.

Рисунок 3.6 - Кнопка “Add”



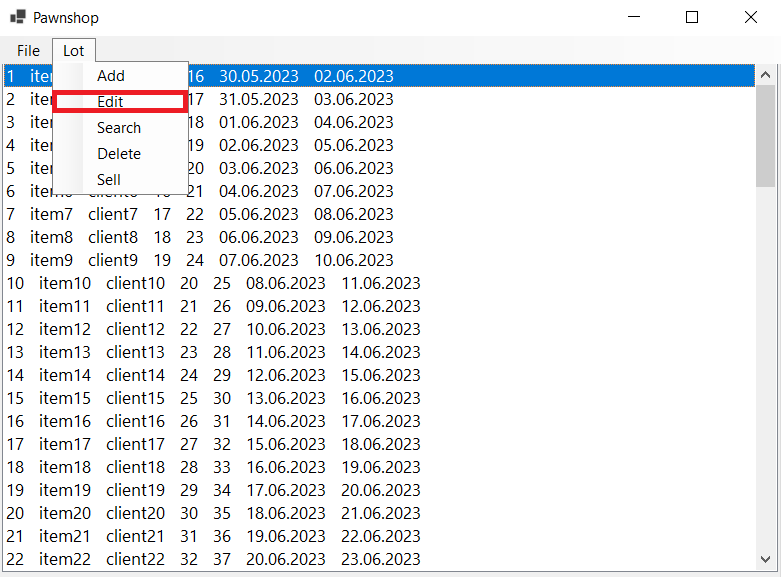
  
Рисунок 3.7 – Форма додавання лотів

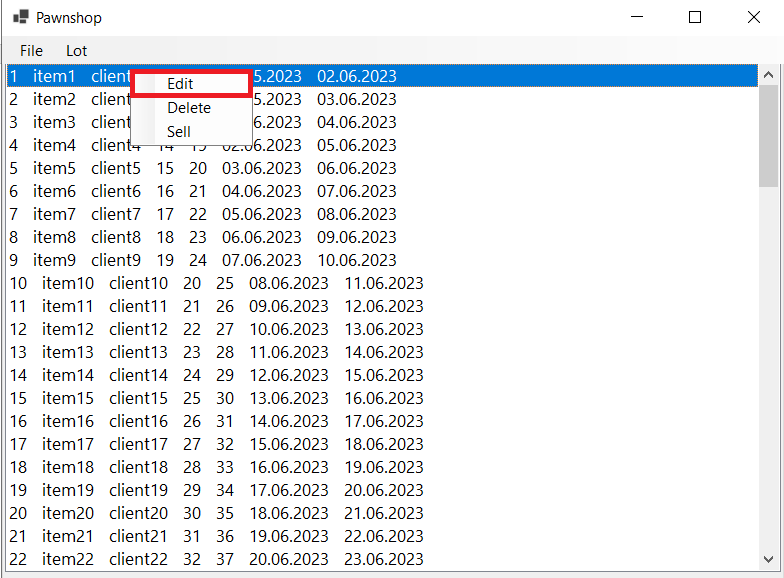
**3.7 Редагування лоту**

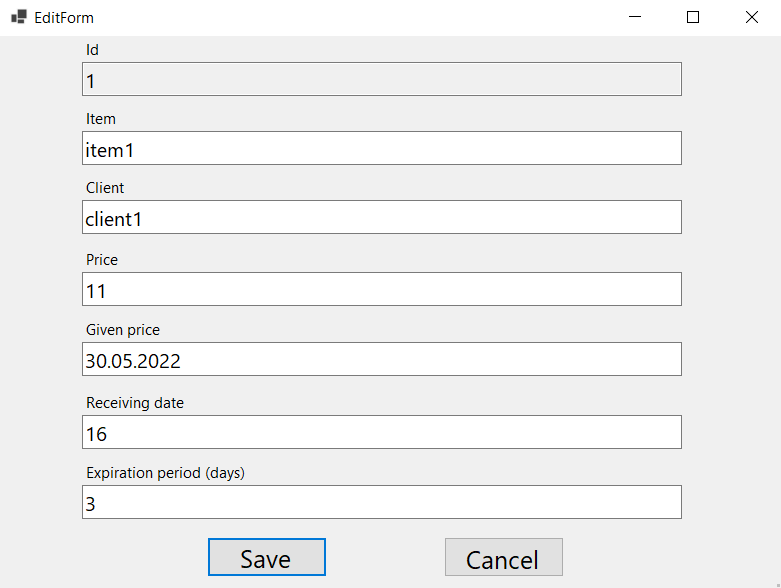
Для редагування лоту користувач може використати два способи:

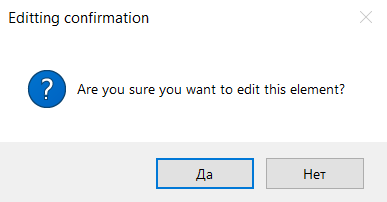
1. Послідовність дій користувача:
2. Користувач обирає потрібний лот та натискає кнопку “Edit” (рис. 3.8) у випадаючому меню у вкладці “Lot”.
3. Відривається форма редагування лоту(рис. 3.10).
4. У формі редагування лоту (рис. 3.10) користувач може редагувати дані лоту: назву предмета, власника, оціночну вартість, видану суму, дату отримання(у форматі “dd/mm/yyyy”) та термін зберігання.
5. Користувач натискає кнопку “Save”.
6. Користувач підтверджує редагування (рис. 3.11).
7. Послідовність дій користувача:
8. Користувач натискає кнопку “Edit” при натисканні правою кнопкою на конкретний лот(рис. 3.9).
9. Відривається форма редагування лоту(рис. 3.10).
10. У формі редагування лоту (рис. 3.10) користувач може редагувати дані лоту: назву предмета, власника, оціночну вартість, видану суму, дату отримання(у форматі “dd/mm/yyyy”) та термін зберігання.
11. Користувач натискає кнопку “Save”.
12. Користувач підтверджує редагування (рис. 3.11).

У результаті програма редагує обраний лот і встановлює відповідні дані, які ввів користувач.

  
Рисунок 3.8 – Кнопка “Edit” у випадаючому меню у вкладці “Lot”

  
Рисунок 3.9 – Кнопка “Edit” при натисканні на конкретний лот

  
Рисунок 3.10 – Форма редагування лоту

  
Рисунок 3.11 – Вікно підтвердження редагування

**3.8 Видалення лоту**

Для видалення лоту користувач може використати два способи:

1. Послідовність дій користувача:
2. Користувач обирає потрібний лот та натискає кнопку “Delete” (рис. 3.12) у спливаючому меню у вкладці “Lot”.
3. Користувач підтверджує свій вибір щодо видалення обраного елемента (рис. 3.14).
4. Послідовність дій користувача:
5. Користувач натискає кнопку “Delete” (рис. 3.13) при натисканні правою кнопкою на потрібний лот.
6. Користувач підтверджує свій вибір щодо видалення обраного елемента (рис. 3.14).

У результаті якщо користувач підтвердив видалення лоту програма видаляє обраний лот зі списку. Якщо не було підтвердження програма повертається на головну сторінку.

Рисунок 3.12 - Кнопка “Delete” у спливаючому меню у вкладці “Lot”

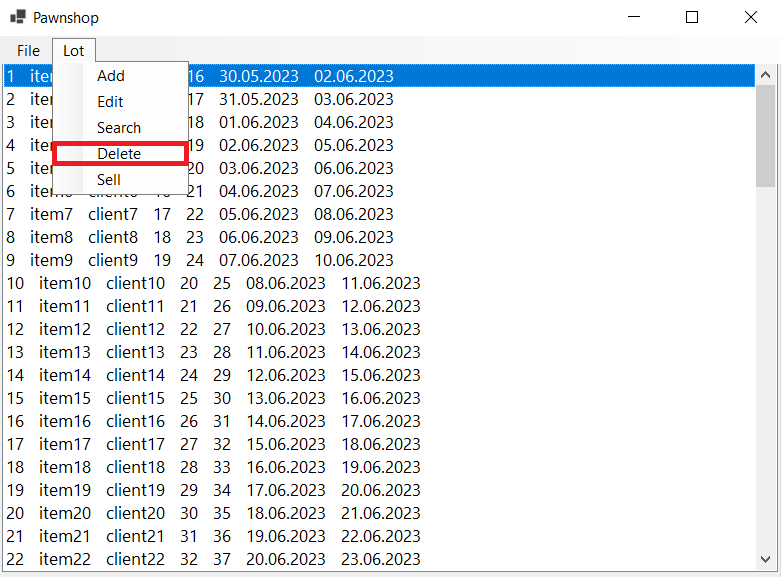
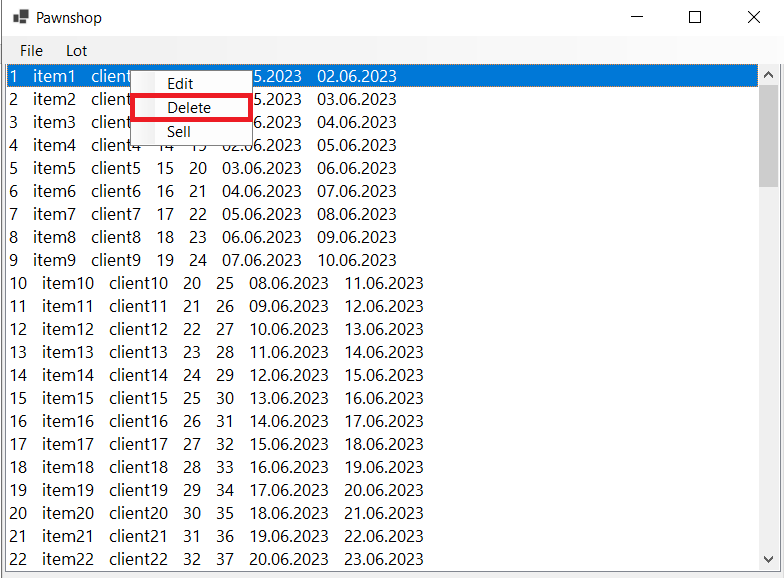
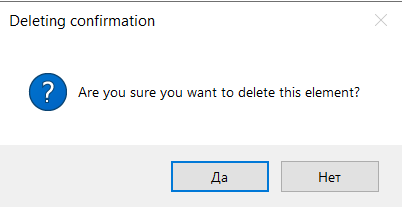


Рисунок 3.13 - Кнопка “Delete” при натисканні на конкретний лот



  
Рисунок 3.14 – Вікно підтвердження видалення лоту

**3.9 Продаж лоту**

Обов’язковою умовою для продажу лоту є вичерпання терміну зберігання, якщо ця умова не дотримана користувачу буде відображена помилка(рис. 3.18).

Для продажу лоту користувач може використати два способи:

1. Послідовність дій користувача:
2. Користувач обирає потрібний лот та натискає кнопку “Sell” (рис. 3.15) у спливаючому меню у вкладці “Lot”.
3. Користувач підтверджує свій вибір щодо продажу обраного елемента (рис. 3.17).
4. Послідовність дій користувача:
5. Користувач натискає кнопку “Sell” (рис. 3.16) при натисканні правою кнопкою на потрібний лот.
6. Користувач підтверджує свій вибір щодо продажу обраного елемента (рис. 3.17).

У результаті якщо користувач підтвердив продаж лоту програма видаляє обраний лот зі списку. Якщо не було підтвердження програма повертається на головну сторінку.

Рисунок 3.15 - Кнопка “Sell” у спливаючому меню у вкладці “Lot”

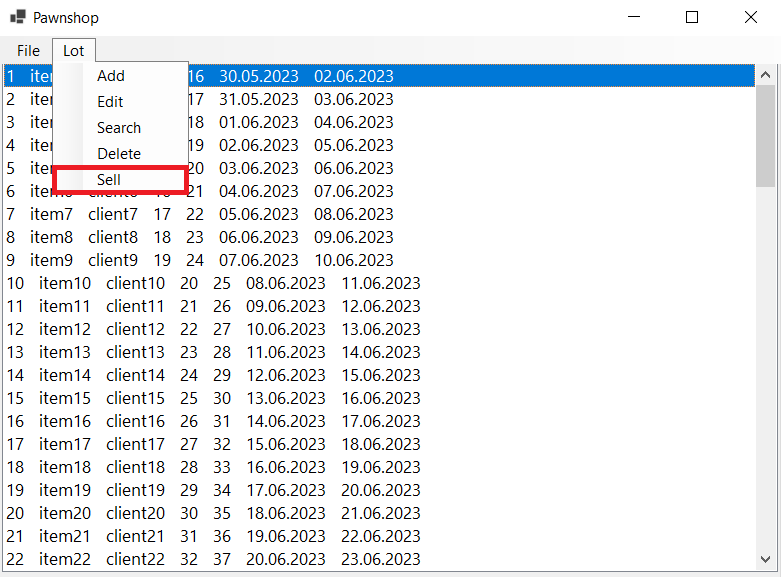
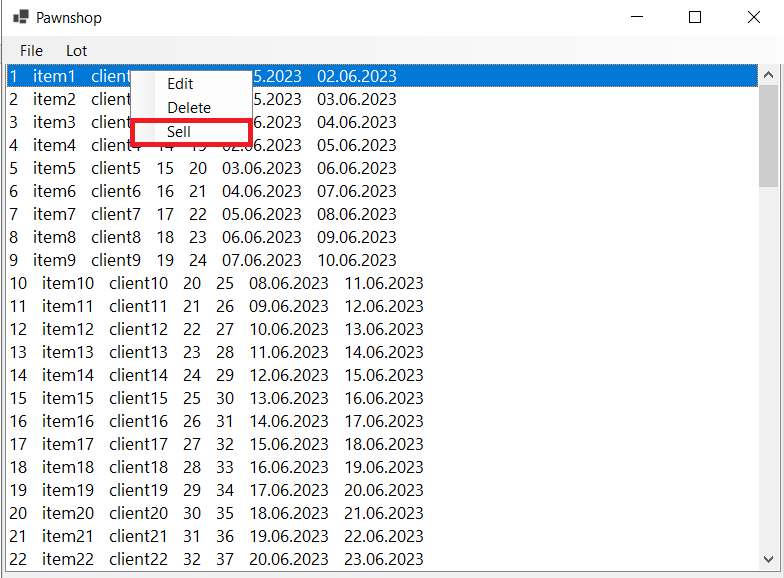
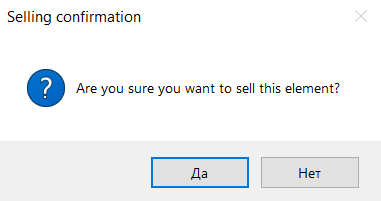
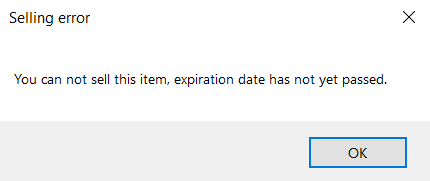


Рисунок 3.16 - Кнопка “Sell” при натисканні на конкретний лот



  
Рисунок 3.17 – Вікно підтвердження продажу лоту

  
Рисунок 3.18 – Вікно сповіщення про помилку недотримання умов для продажу лотів

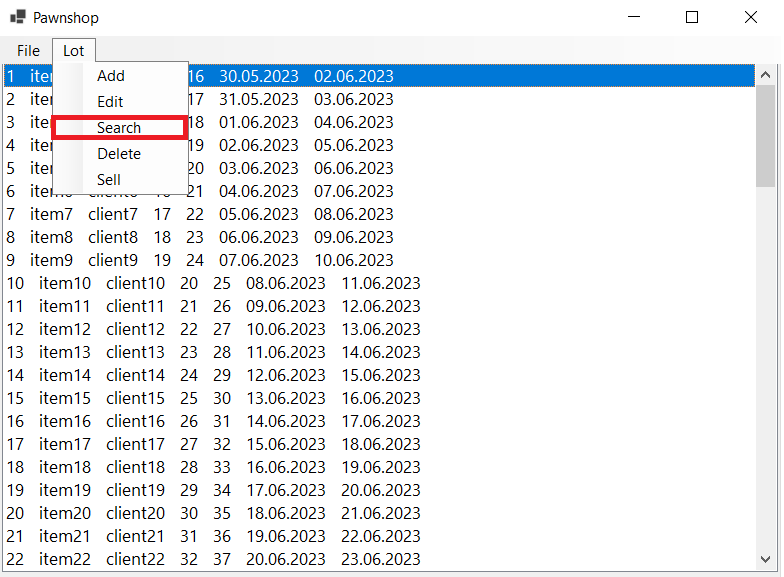
**3.10 Пошук лотів**

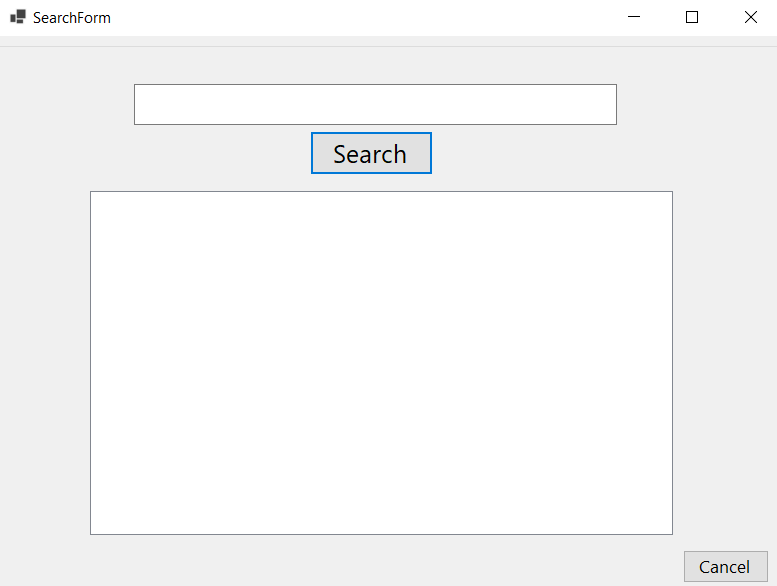
Послідовність дій користувача:

1. Користувач натискає кнопку “Search” (рис. 3.19) у спливаючому меню у вкладці “Lot”.
2. Відривається форма пошуку лотів(рис. 3.20).
3. У відповідне поле користувач може вводити будь-які дані за якими буде проведений пошук по всім параметрам лоту.
4. Для отримання результатів пошуку користувач натискає на кнопку ”Search”.

У результаті програма проводить пошук за отриманими від користувача даними та виводить результат пошуку у відповідне поле у формі пошуку.

Рисунок 3.19 - Кнопка “Search” у спливаючому меню у вкладці “Lot”



  
Рисунок 3.20 – Форма пошуку лотів

**Висновки**

За своїми функціональними можливостями програма повністю відповідає заявленим вимогам. Всі необхідні функції реалізовані та доведені до робочого стану. Стан програми можна вважати завершеною, але все одно залишається багато шляхів для вдосконалення та розвитку. Наприклад, додавання нових функцій, нових типів даних, що можуть відображатися програмою, вдосконалення старих і т.д.

В процесі роботи над проектом я отримав чудовий досвід у створенні Windows Forms Application та роботи з невеликими локальними файловими базами даних, покращив свої знання мови C#, навчився використовувати репозиторії GitHub.

**ДОДАТОК А**

namespace Pawnshop

{

internal static class Program

{

[STAThread]

static void Main()

{

ApplicationConfiguration.Initialize();

Application.Run(new MainForm());

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.Json;

using System.Text.Json.Serialization;

using System.Threading.Tasks;

namespace Pawnshop.Models

{

public class Lot

{

public int id { get; set; }

public string item { get; set; }

public string client { get; set; }

public double price { get; set; }

public double price\_given { get; set; }

public DateTime date { get; set; }

public DateTime date\_expire { get; set;}

[JsonConstructor]

public Lot(int id, string item, string client, double price, double price\_given, DateTime date, DateTime date\_expire)

{

this.id = id;

this.item = item;

this.client = client;

this.price = price;

this.price\_given = price\_given;

this.date = date;

this.date\_expire = date\_expire;

}

public override string ToString()

{

return $"{id} {item} {client} {price} {price\_given} {date.ToShortDateString()} {date\_expire.ToShortDateString()}";

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Pawnshop.Models

{

public class PawnshopList

{

public int IdCounter { get; set; }

public bool IsDirty { set; get; }

public List<Lot> Lots { set; get; }

public PawnshopList()

{

IsDirty = false;

Lots = new List<Lot>();

IdCounter = 1;

}

}

}

using Pawnshop.Models;

using Pawnshop.Forms;

using Pawnshop.Data;

using System.Globalization;

using static System.Reflection.Metadata.BlobBuilder;

using System.Windows.Forms;

using System.Diagnostics;

using System.Diagnostics.Metrics;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Window;

namespace Pawnshop

{

public partial class MainForm : Form

{

PawnshopList pawnshop;

public static MainForm link;

public static PawnshopList pawnshopMain;

public static PawnshopList GetMain()

{

if (pawnshopMain == null)

{

pawnshopMain = new PawnshopList();

}

return pawnshopMain;

}

public MainForm()

{

InitializeComponent();

link = this;

this.FormClosing += MainForm\_FormClosing;

pawnshop = GetMain();

mainFormBindingSource.DataSource = pawnshop.Lots;

bool state = DataAccess.stateLoad();

if(state)

{

DataAccess.Load(pawnshop);

mainFormBindingSource.ResetBindings(true);

DataAccess.stateSave(false);

}

}

public void AddNewLot(string item, string client, double price, double price\_given, int expiration\_period)

{

int id = pawnshop.IdCounter++;

Lot newLot = new Lot(id, item, client, price, price\_given, DateTime.Now, DateTime.Now.AddDays(expiration\_period));

pawnshop.Lots.Add(newLot);

pawnshop.IsDirty = true;

mainFormBindingSource.ResetBindings(true);

}

public void EditLot(int id, string item, string client, double price, double price\_given, DateTime date, int expiration\_period)

{

Lot toBeChanged = pawnshop.Lots.Find(x => x.id == id);

if (toBeChanged != null)

{

toBeChanged.item = item;

toBeChanged.price = price;

toBeChanged.client = client;

toBeChanged.price = price;

toBeChanged.price\_given = price\_given;

toBeChanged.date = date;

toBeChanged.date\_expire = date.AddDays(expiration\_period);

pawnshop.IsDirty = true;

mainFormBindingSource.ResetBindings(true);

}

}

public List<Lot> searchFor(string searchValue)

{

var searchResults = new List<Lot>();

int searchInt;

bool isInt = int.TryParse(searchValue, out searchInt);

double searchDouble;

bool isDouble = double.TryParse(searchValue, out searchDouble);

DateTime searchDate;

bool isDate = DateTime.TryParseExact(searchValue, "dd.MM.yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out searchDate);

searchResults = pawnshop.Lots.Where(obj =>

(isInt && obj.id == searchInt) ||

obj.item == searchValue ||

obj.client == searchValue ||

(isDouble && (obj.price == searchDouble || obj.price\_given == searchDouble)) ||

(isDate && (obj.date.ToShortDateString() == searchValue || obj.date\_expire.ToShortDateString() == searchValue))

).Distinct().ToList();

return searchResults;

}

private void testDataToolStripMenuItem\_Click(object sender, EventArgs e)

{

int n = 10;

for (int i = 1; i < n+1; i++)

{

if (i % 2 == 0)

{

var newLot = new Lot(pawnshop.IdCounter, "item" + i, "client" + i, 10 + i, 15 + i, DateTime.Now.AddDays(i), DateTime.Now.AddDays(i + 3));

pawnshop.IdCounter++;

pawnshop.Lots.Add(newLot);

}

else

{

var newLot = new Lot(pawnshop.IdCounter, "item" + i, "client" + i, 10 + i, 15 + i, DateTime.Now.AddDays(-365+i), DateTime.Now.AddDays(-365 + i + 3));

pawnshop.IdCounter++;

pawnshop.Lots.Add(newLot);

}

}

mainFormBindingSource.ResetBindings(true);

mainFormBindingSource.DataSource = pawnshop.Lots;

}

private void exitToolStripMenuItem\_Click(object sender, EventArgs e)

{

safeExit();

}

private void addToolStripMenuItem\_Click(object sender, EventArgs e)

{

var AddForm = new AddForm();

AddForm.ShowDialog();

}

private void ediToolStripMenuItem\_Click(object sender, EventArgs e)

{

Lot selectedItem = (Lot)ItemList.SelectedItem;

if (selectedItem != null)

{

var EditForm = new EditForm();

EditForm.id(selectedItem);

EditForm.ShowDialog();

}

else

{

string message = "Catched: System.NullReferenceException. You are trying to edit nothing.";

string caption = "Error";

var result = MessageBox.Show(message, caption, MessageBoxButtons.OK, MessageBoxIcon.Question);

}

}

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

{

safeExit();

}

private void clearToolStripMenuItem\_Click(object sender, EventArgs e)

{

pawnshop.Lots.Clear();

pawnshop.IdCounter = 1;

mainFormBindingSource.ResetBindings(true);

}

private void saveToolStripMenuItem\_Click(object sender, EventArgs e)

{

DataAccess.Save(pawnshop);

}

private void loadToolStripMenuItem\_Click(object sender, EventArgs e)

{

DataAccess.Load(pawnshop);

mainFormBindingSource.ResetBindings(true);

}

private void ItemList\_MouseDown(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Right)

{

int index = ItemList.IndexFromPoint(e.Location);

if (index != ListBox.NoMatches)

{

ItemList.SelectedIndex = index;

contextMenuStrip1.Show(ItemList, e.Location);

}

}

}

private void editToolStripMenuItem\_Click(object sender, EventArgs e)

{

var EditForm = new EditForm();

Lot selectedItem = (Lot)ItemList.SelectedItem;

EditForm.id(selectedItem);

EditForm.ShowDialog();

}

private void deleteToolStripMenuItem1\_Click(object sender, EventArgs e)

{

Lot selectedItem = (Lot)ItemList.SelectedItem;

delete(selectedItem);

}

private void deleteToolStripMenuItem\_Click(object sender, EventArgs e)

{

Lot selectedItem = (Lot)ItemList.SelectedItem;

delete(selectedItem);

}

public void delete(Lot selectedItem)

{

if (selectedItem != null)

{

string message = "Are you sure you want to delete this element?";

string caption = "Deleting confirmation";

var result = MessageBox.Show(message, caption, MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

pawnshop.Lots.Remove(selectedItem);

pawnshop.IsDirty = true;

mainFormBindingSource.ResetBindings(true);

}

}

else

{

string message = "Catched: System.NullReferenceException. You are trying to delete nothing.";

string caption = "Error";

var result = MessageBox.Show(message, caption, MessageBoxButtons.OK);

}

}

public void safeExit()

{

if (pawnshop.IsDirty)

{

string message = "Do you want to save the document?";

string caption = "Saving confirmation";

var result = MessageBox.Show(message, caption, MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

DataAccess.Save(pawnshop);

}

else

{

DataAccess.Load(pawnshop);

}

DataAccess.stateSave(true);

Environment.Exit(0);

}

else

{

DataAccess.stateSave(true);

Environment.Exit(0);

}

}

private void searchToolStripMenuItem\_Click(object sender, EventArgs e)

{

this.Hide();

var SearchForm = new SearchForm();

SearchForm.Show();

}

private void sellToolStripMenuItem\_Click(object sender, EventArgs e)

{

Lot selectedItem = (Lot)ItemList.SelectedItem;

if(selectedItem.date\_expire < DateTime.Now)

{

sell(selectedItem);

}

else

{

string message = "You can not sell this item, expiration date has not yet passed.";

string caption = "Selling error";

var res = MessageBox.Show(message, caption, MessageBoxButtons.OK);

}

}

public void sell(Lot selectedItem)

{

if (selectedItem != null)

{

string message = "Are you sure you want to sell this element?";

string caption = "Selling confirmation";

var result = MessageBox.Show(message, caption, MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

pawnshop.Lots.Remove(selectedItem);

pawnshop.IsDirty = true;

mainFormBindingSource.ResetBindings(true);

}

}

else

{

string message = "Catched: System.NullReferenceException. You are trying to sell nothing.";

string caption = "Error";

var result = MessageBox.Show(message, caption, MessageBoxButtons.OK);

}

}

private void sellToolStripMenuItem1\_Click(object sender, EventArgs e)

{

Lot selectedItem = (Lot)ItemList.SelectedItem;

if (selectedItem.date\_expire < DateTime.Now)

{

sell(selectedItem);

}

else

{

string message = "You can not sell this item, expiration date has not yet passed.";

string caption = "Selling error";

var res = MessageBox.Show(message, caption, MessageBoxButtons.OK);

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Pawnshop.Models;

using Pawnshop.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using Pawnshop.Data;

namespace Pawnshop.Forms

{

public partial class AddForm : Form

{

public AddForm()

{

InitializeComponent();

}

private void AddForm\_FormClosing\_1(object sender, FormClosingEventArgs e)

{

this.Hide();

}

private void submit\_button\_Click(object sender, EventArgs e)

{

{

string item = item\_box.Text;

string itemNoSpaces = item.Replace(" ", string.Empty);

if ((string.IsNullOrEmpty(item)) || (string.IsNullOrEmpty(itemNoSpaces)))

{

MessageBox.Show("Enter the valid value for the item.");

return;

}

string client = client\_box.Text;

string clientNoSpaces = client.Replace(" ", string.Empty);

if ((string.IsNullOrEmpty(client)) || (string.IsNullOrEmpty(clientNoSpaces)))

{

MessageBox.Show("Enter the valid value for the client.");

return;

}

if ((!double.TryParse(price\_box.Text, out double price)) || (price < 0))

{

MessageBox.Show("Enter the valid value for the price.");

return;

}

if ((!double.TryParse(given\_price\_box.Text, out double price\_given)) || (price\_given < 0))

{

MessageBox.Show("Enter the valid value for the given price.");

return;

}

if ((!int.TryParse(period\_box.Text, out int expiration\_period)) || (expiration\_period < 0))

{

MessageBox.Show("Enter the valid value for the expiration period.");

return;

}

this.Hide();

MainForm.link.AddNewLot(item, client, price, price\_given, expiration\_period);

}

}

private void button1\_Click\_1(object sender, EventArgs e)

{

this.Hide();

}

}

}

using Pawnshop.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Runtime.InteropServices.JavaScript.JSType;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Header;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.ListView;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Window;

namespace Pawnshop.Forms

{

public partial class EditForm : Form

{

public int from;

public EditForm()

{

InitializeComponent();

if (from != 1)

{

from = 0;

}

}

private void button2\_Click(object sender, EventArgs e)

{

string item = item\_box.Text;

string itemNoSpaces = item.Replace(" ", string.Empty);

if ((string.IsNullOrEmpty(item)) || (string.IsNullOrEmpty(itemNoSpaces)))

{

MessageBox.Show("Invalid item name.");

return;

}

string client = client\_box.Text;

string clientNoSpaces = client.Replace(" ", string.Empty);

if ((string.IsNullOrEmpty(client)) || (string.IsNullOrEmpty(clientNoSpaces)))

{

MessageBox.Show("Invalid client name.");

return;

}

if ((!double.TryParse(price\_box.Text, out double price)) || (price < 0))

{

MessageBox.Show("Invalid price.");

return;

}

if ((!double.TryParse(given\_price\_box.Text, out double price\_given)) || (price\_given < 0))

{

MessageBox.Show("Invalid given price.");

return;

}

string dateString = recieving\_box.Text;

string[] formats = { "dd.MM.yyyy" };

DateTime date;

if (!DateTime.TryParseExact(dateString, formats, CultureInfo.InvariantCulture, DateTimeStyles.None, out date))

{

MessageBox.Show("Invalid receiving date.");

return;

}

if ((!int.TryParse(period\_box.Text, out int expiration\_period)) || (expiration\_period < 0))

{

MessageBox.Show("Invalid expiration period.");

return;

}

int id = int.Parse(id\_box.Text);

string message = "Are you sure you want to edit this element?";

string caption = "Editting confirmation";

var result = MessageBox.Show(message, caption, MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

MainForm.link.EditLot(id, item, client, price, price\_given, date, expiration\_period);

this.Hide();

}

}

private void button1\_Click(object sender, EventArgs e)

{

this.Hide();

}

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

{

this.Hide();

}

public void id(Lot changed)

{

id\_box.Text = changed.id.ToString();

item\_box.Text = changed.item;

client\_box.Text = changed.client;

price\_box.Text = changed.price.ToString();

given\_price\_box.Text = changed.price\_given.ToString();

recieving\_box.Text = changed.date.ToShortDateString();

period\_box.Text = 3.ToString();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Pawnshop.Models;

using Pawnshop.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Header;

namespace Pawnshop.Forms

{

public partial class SearchForm : Form

{

public static string lastText;

public SearchForm()

{

InitializeComponent();

search\_input.Text = lastText;

var res = MainForm.link.searchFor(lastText);

lotBindingSource.DataSource = res;

}

private void Cancel\_Click(object sender, EventArgs e)

{

this.Hide();

MainForm.link.Show();

}

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

{

this.Hide();

MainForm.link.Show();

}

private void search\_button\_Click(object sender, EventArgs e)

{

var str = search\_input.Text;

search(str);

}

private void editToolStripMenuItem\_Click(object sender, EventArgs e)

{

Lot selectedItem = (Lot)search\_output.SelectedItem;

lastText = search\_input.Text;

var EditForm = new EditForm();

EditForm.from = 1;

EditForm.id(selectedItem);

EditForm.ShowDialog();

var str = search\_input.Text;

search(str);

}

private void deleteToolStripMenuItem\_Click(object sender, EventArgs e)

{

Lot selectedItem = (Lot)search\_output.SelectedItem;

MainForm.link.delete(selectedItem);

var res = MainForm.link.searchFor(lastText);

lotBindingSource.DataSource = res;

}

public void search(string str)

{

lastText = str;

var res = MainForm.link.searchFor(str);

lotBindingSource.DataSource = res;

if(res.Count == 0)

{

string message = "No lots found.";

var result = MessageBox.Show(message, "", MessageBoxButtons.OK);

}

}

private void sellToolStripMenuItem\_Click(object sender, EventArgs e)

{

Lot selectedItem = (Lot)search\_output.SelectedItem;

if (selectedItem.date\_expire < DateTime.Now)

{

MainForm.link.sell(selectedItem);

}

else

{

string message = "You can not sell this item, expiration date has not yet passed.";

string caption = "Selling error";

var result = MessageBox.Show(message, caption, MessageBoxButtons.OK);

}

var res = MainForm.link.searchFor(lastText);

lotBindingSource.DataSource = res;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Text.Json;

using System.Threading.Tasks;

using Pawnshop.Models;

namespace Pawnshop.Data

{

internal static class DataAccess

{

const string DATA\_PATH = "Data\\Pawnshop.json";

const string DATA\_PATH2 = "Data\\State.json";

public static void Save(PawnshopList pawnshop)

{

string jsonString = JsonSerializer.Serialize(pawnshop);

File.WriteAllText(DATA\_PATH, jsonString);

pawnshop.IsDirty = false;

}

public static void Load(PawnshopList pawnshop)

{

string jsonString = File.ReadAllText(DATA\_PATH);

var newPawnshop = JsonSerializer.Deserialize<PawnshopList>(jsonString);

pawnshop.Lots.Clear();

pawnshop.Lots.AddRange(newPawnshop.Lots);

pawnshop.IdCounter = newPawnshop.IdCounter;

pawnshop.IsDirty = false;

}

public static bool stateLoad()

{

string jsonString = File.ReadAllText(DATA\_PATH2);

var newState = JsonSerializer.Deserialize<bool>(jsonString);

return newState;

}

public static void stateSave(bool IsLoaded)

{

string jsonString = JsonSerializer.Serialize(IsLoaded);

File.WriteAllText(DATA\_PATH2, jsonString);

}

}

}