Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліні “ Об’єктно -орієнтоване програмування”

Керівник , доцент Бондарєв В. М.

Студент гр. ПЗПІ-19-2 Гаврюшенко А. А.

Комісія:

Ст. викл.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Черепанова Ю.Ю.,

Проф. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бондарєв В. М.

Доц. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Побіженко І. О.

Харків 2020

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра: ***Програмної інженерії***

Дисципліна: ***Об’єктно-орієнтоване програмування***

Спеціальність: ***121***  ***Інженерія програмного забезпечення***

Освітня програма: ***Програмна інженерія***

Курс 1 .Група *ПЗПІ-19 -2* . Семестр *2* .

***ЗАВДАННЯ***

***на курсовий проект студента***

*Гаврюшенка Артема Анатолійовича**\_*

1 Тема проекту: *Ломбард*

2 Термін здачі студентом закінченого проекту: ***“31” - травня - 2020 р***.

3 Вихідні дані до проекту:

*Специфікація програми, методичні вказівки до виконання курсової роботи.*

4 Зміст розрахунково-пояснювальної записки:

*Вступ,* *специфікація програми, проектна специфікація, інструкція користувача, висновки \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5 Перелік графічного матеріалу:

*Схема об’єктної моделі, алгоритми, приклади екранних форм* \_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 21-02-2020 р. |
| 2 | Формулювання вимог до програми | 01-03-2020 – 31-03-2020 р. |
| 3 | Розробка підсистеми зберігання та додавання товарів. | 02-04-2020 – 12-04-2020 р. |
| 4 | Розробка підсистеми зберігання та реєстрації клієнтів. | 12-04-2020 – 22-04-2020 р. |
| 5 | Розробка функцій продажу та повернення товарів. | 24-04-2020 – 30-04-2020 р. |
| 6 | Розробка функцій зберігання та завантаження даних. | 05-05-2020 – 11-05-2020 р. |
| 7 | Тестування і доопрацювання розробленої програмної системи. | 10-05-2020 – 15-05-2020 р. |
| 8 | Оформлення пояснювальної записки, додатків, графічного матеріалу | 25-05-2020 – 05-06-2020 р. |
| 9 | Захист | 01-06-2020 – 19-06-2020 р. |

Студент Гаврюшенко А. А.

Керівник Бондарєв В. М.

« 21 »\_лютого\_\_\_\_\_\_\_\_\_\_ 2020 р.

**РЕФЕРАТ**

Пояснювальна записка до курсової роботи: 38 с., 21 рис., 1 додаток, 6 джерел.

МОВА ПРОГРАМУВАННЯ С#, WINDOWS FORMS, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ЛОМБАРД, НЕРУХОМІСТЬ, КУПІВЛЯ, ПРОДАЖ, КЛІЄНТ, АДМІНІСТРАТОР.

Метою роботи є розробка програмного забезпечення “Ломбард”, базуючись на принципах об’єктно-орієнтованого програмування.

Інструменти, що використовувались для реалізації завдання: Microsoft Visual Studio 2019, Windows Forms, платформа .NET Framework, мови програмування C#.

В результаті створено програму, що має назву «Ломбард», яка є програмним забезпеченням для закладів-ломбардів. Програма має два додатки: для адміністратора та клієнтів. Реалізовані наступні функції: реєстрація нового клієнту в системі, перегляд існуючих продуктів, заповнення заявки для продажу товару/нерухомості, викупу предметів, покупки доступних предметів. Для адміністратора додаються функції редагування/видалення позицій з бази, затвердження або відхилення заявок. Реалізований запис даних до файлів.

ЗМІСТ

[Вступ 6](#_Toc532289188)

[1 Специфікація програми 7](#_Toc532289189)

[1.1 Інтерфейс користувача 7](#_Toc532289190)

[1.2 Окремі функції користувача](#_Toc532289191) 10

[2 Проектна специфікація 1](#_Toc532289192)4

[2.1 Об’єктна модель програми 1](#_Toc532289193)4

[2.2 Структура проекту  1](#_Toc532289194)4

[2.3 Об’єктна модель 1](#_Toc532289194)5

[2.4 Збереження та доступ до даних. Контроль версій 1](#_Toc532289194)9

[3 Інструкція користувача 2](#_Toc532289195)0

[3.1 Запуск програми  2](#_Toc532289194)0

[3.2 Робота програми  2](#_Toc532289194)1

[Висновки 2](#_Toc532289196)8

[Перелік джерел посилання 2](#_Toc532289196)9

[Додаток А Код програми 3](#_Toc532289197)0

ВСТУП

Метою даного курсового проекту є створення спеціалізованого програмного забезпечення для ломбарду. Реалізація проекту ґрунтується на принципах об’єктно-орієнтованого програмування, тому для роботи була обрана така мова програмування як С# та середа розробки Visual Studio. За задачу було також поставлено створення графічного інтерфейсу користувача, що потребувало використання додаткового інструменту – Windows Forms.

Галуззю застосування є заклади-ломбарди та заклади, що орієнтовані на продаж нерухомості, оскільки програма, розроблена у цьому проекті, може оптимізувати та значно полегшити їх роботу. Дане програмне забезпечення пропонує: централізоване зберігання інформації про товари та нерухомість, взаємодію з клієнтами, зручний сервіс для клієнтів. Даний проект являє собою готовий для використання продукт, але також може бути відредагований згідно потреб закладу і може бути використаний для всіх типів ломбардів: ювелірних, битових, антикварних та інших. Така гнучкість робить галузь застосування надзвичайно широкою та перспективною.

Діяльність ломбардів особливо потребує чіткого та автоматизованого інструменту для зберігання та управління інформацією, тому створення такого програмного забезпечення є вкрай актуальним та корисним. Також цей проект може стати основою та прототипом для створення більш масштабних систем – наприклад, програмного забезпечення для мережі ломбардів, або інших закладів, що функціонують на подібних засадах.

**1 СПЕЦИФІКАЦІЯ ПРОГРАМИ**

* 1. Інтерфейс користувача

Для клієнтської версії додатку інтерфейс користувача складається з 6 вікон. При запуску програми з’являється перше - вікно входу. Можливий перехід на форму для реєстрації у системі. (Див. рис. 1.1 та 1.2)

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 1.1 – Вікно входу

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 1.2 – Вікно реєстрації

Наступним вікном інтерфейсу користувача є основне вікно програми.

(Див. рис. 1.3)

З нього можливий перехід на наступні вікна:

1. Корзина клієнта

б) Форма заповнення заявки на продаж (див. рис. 1.4)

в) Список схвалених заявок 5

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 1.3 – Основне вікно

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 1.4 – Формування заявки

Для адміністраторської версії додатку інтерфейс складається з двох вікон. При запуску програми відкривається перше – основне вікно програми. (Див. рис. 1.5)

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 1.5 – Вікно адміністратора

З нього можливий перехід на друге вікно – перегляд заявок, відправлених користувачами. (Див. рис. 1.6)

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 1.6 – Вікно заявок

* 1. Опис окремих функцій

Користувачу доступні такі функції:

* + 1. Вхід до системи за логіном та паролем

Клієнт входить в програму за допомогою логіна і пароля.

Імена і паролі клієнтів зберігаються в даних програми.

При правильному введенні пароля клієнт потрапляє в меню клієнта.

Основний сценарій

1. Клієнт потрапляє на форму з авторизацією (рис. 1).

2. Клієнт вводить логін і пароль і натискає кнопку "OK".

3. Якщо дані введені вірно - відбувається перехід в меню клієнта.

Додатковий сценарій:

1. Якщо дані введені невірно - на формі введення з'являється повідомлення про помилку і можливий повторний вхід. Число спроб не обмежена.

2. Якщо користувач не зареєстрований, він може вибрати вікно «Реєстрація».

* + 1. Реєстрація у системі за логіном, паролем та номером паспорту

Покупець реєструється і вводить свої дані.

Основний сценарій:

1. Користувач вибирає вікно «Registration».

2. Відкривається форма для заповнення даних про покупця.

3. Користувач заповнює форму.

4. Натискає кнопку «register».

* + 1. Перегляд наявних у базі товарів

Про товари надається інформація: назва, фотографія, логін продавця, опис, ціна (вартість розраховується у відповідності із терміном зберігання товару у закладі; після закінчення терміну зберігання ціна піднімається на п’яту частину оціночної вартості).

* + 1. Продаж товару

Для цього необхідно сформувати заявку: вказати

назву, опис, тип товару, додати фотографію. Заявка відсилається на перевірку адміністратору. Клієнт вибирає пункт продажу. Клієнт формує заявку на здачу товару в ломбард. Заповнює назву і фото, потім відправляє на оцінку.

сценарій:

1. Вибирає пункт «Sell».

2. Відкривається форма подачі даних.

3. Покупець заповнює поля форми.

4. Натискає кнопку «Send».

* + 1. Викуп товару

Можливість викупити товар, що був схвалений адміністратором та

виставлений на продаж. Викуп дозволений протягом терміну зберігання зазначеного адміністратором, проте кожен день ціна зростає. Після цього терміну товар стає доступним для всіх клієнтів.

* + 1. Перегляд історії придбань.

Користувач вибирає кнопку «Purchase». Відкривається нове вікно, у якому представлені придбані раніше товари.

* + 1. Збереження даних

У процесі роботи користувач змінює дані, які знаходяться в оперативній пам'яті. Якщо дані змінені, але ще не введено постійної пам'яті, то перед закриттям програми користувач отримує пропозицію зберегти дані.

Адміністратору доступні такі функції:

* + 1. Перегляд товарів у базі. Можливість редагування даних: назви, опису, вартості та фотографії

1. На адмінпанелі вибирає товар із загального переліку.

2. Натискає кнопку edit.

3. Відкривається форма зміни товару. Поля форми заповнені.

4. Змінює поля форми, змінює файл зображення.

5. Натискає кнопку «Зберегти».

* + 1. Додавання товарів до бази

1. На адмінпанелі натискає кнопку Add.

2. Відкривається форма. Змінює поля форми, змінює файл зображення.

3. Натискає кнопку «Зберегти».

* + 1. Видалення одного або декілька товарів з бази

1. Адміністратор вибирає товар із загального переліку на адмінпанелі.

2. Натискає кнопку Del.

3. Відкривається панель із запитом на видалення.

4. Після отримання підтвердження, товар видаляється, список оновлюється.

* + 1. Перегляд заявок від користувачів. Можливість внесення до них змін. Схвалення заявки або відхилення

1. Адміністратор вибирає вкладку «Applications»

2. Відкривається панель обробки нових заявок

* + 1. Можливість запису або зчитування інформації з файлу.

Якщо дані змінені, але ще не введено постійної пам'яті, то перед закриттям програми користувач отримує пропозицію зберегти дані.

1. **ПРОЕКТНА СПЕЦИФІКАЦІЯ**

2.1 Архітектура проекту

Програма буде написана на мові С # з використанням середовища розробки Microsoft Visual Studio Community 2019 на платформі .NET Framework 4.7.2, тип програми: настільний з використанням Windows Forms.

Вибір мови С# обумовлюється тим, що на даний момент це один з найбільш розвинених об'єктно-орієнтованих мов, а завдання має на увазі об’єктно-орієнтований підхід. Вона є найбільш досконалою і комфортною у використанні з більшості існуючих мов.

Я вибрав С # .NЕТ Framework серед інших платформ через його розвиненості і зрілості.

Це повнофункціональна платформа з великою кількістю керуючих елементів, яка підтримує розробку сучасних додатків. Явним недоліком є відсутність мультиплатформенності, а саме можливість роботи програми лише тільки на апаратах під управлінням ОС Windows.

Для взаємодії з користувачем був обраний графічний інтерфейс через його зрозумілості та інтуїтивної простоти використання для клієнта. Реалізація відбувається через Windows Forms, які є частиною платформи .NET Framework.

2.2 Структура проекту

За умов та кваліфікаційних вимог, поставлених в даному завданні курсового проекту, максимально ефективно з точки зору зручності програмування і використання, буде створення двох окремих додатків: для клієнта і адміністратора оскільки як їх функціонал не перетинається.

Так само в програму будуть включені набори тестових даних для перевірки коректності її роботи і тестування функцій.

Таким чином модель проекту має наступний вигляд (Див. рис. 2.1)

Клієнт

Адміністратор

Хранилище данних

Модель

Набори тестових даних

Шар доступа до данних

Рисунок 2.1 – Модель проекту

2.3 Об’єктна модель

2.3.1 Опис

Список товарів включає в себе список всіх доступних предметів і квартир.

Адміністратор додає товари, розглядає заявки, переглядає список товарів, вносить зміни, відзначає продані варіанти, видаляє товари з бази.

Клієнт переглядає доступні до покупки товари, додає сподобалися в кошик, здійснює покупку, залишає заявку на продаж товару.

Є два типи товарів: квартири і предмети.

Про квартиру відомо: зображення, адреса, вартість, дата прийому на баланс ломбарду, термін зберігання, данні клієнта, що здав.

Про предмет відомо: найменування, зображення, вартість, дата прийому на баланс ломбарду, термін зберігання, данні клієнта, що здав.

Про адміністратора відомо: ім'я, номер паспорта.

Про покупця відомо: ім'я, номер паспорта, пароль.

До простих типів не належать:

Список квартир, квартира, список предметів, предмет, користувач ( клієнт), корзина, заявка.

З цих типів відокремлюємо ті, які не є колекціями і не містять колекцій:

Заявка, квартира, предмет, клієнт.

Перші призначені для користувача типи:

Предмет - назва, фото, ціна, дата прийому.

Квартира - фото, ціна, адреса, дата прийому.

Покупець - логін, ім'я, номер паспорта, пароль, колекція придбаних товарів.

Решта типів:

Список предметів - колекція предметів.

Список квартир - колекція квартир.

Список користувачів - колекція користувачів.

Заявка – товар, клієнт.

Кошик - колекція обраних до покупки товарів та клієнт.

Ломбард-кілька колекцій: користувачів, квартир, предметів, заявок.

* + 1. Методи

Основні методи:

* Реєстрація нових клієнтів програми (користувач, список користувачів).
* Відправка заявки на продаж товару (користувач, заявка).
* Схвалення заявки (адміністратор, заявка).
* Перегляд списку товарів доступних до покупки (клієнт, список доступних товарів).
* Купівля обраних товарів (покупець, корзина, список товарів).
* Додавання нового товару (адміністратор, список квартир, квартира, список предметів, предмет).
* Зміна товарів (адміністратор, список квартир, квартира, список предметів, предмет).
* Видалення товарів (адміністратор, список квартир, квартира, список предметів, предмет).

Зв’язки між класами представлені на рисунку 2.2

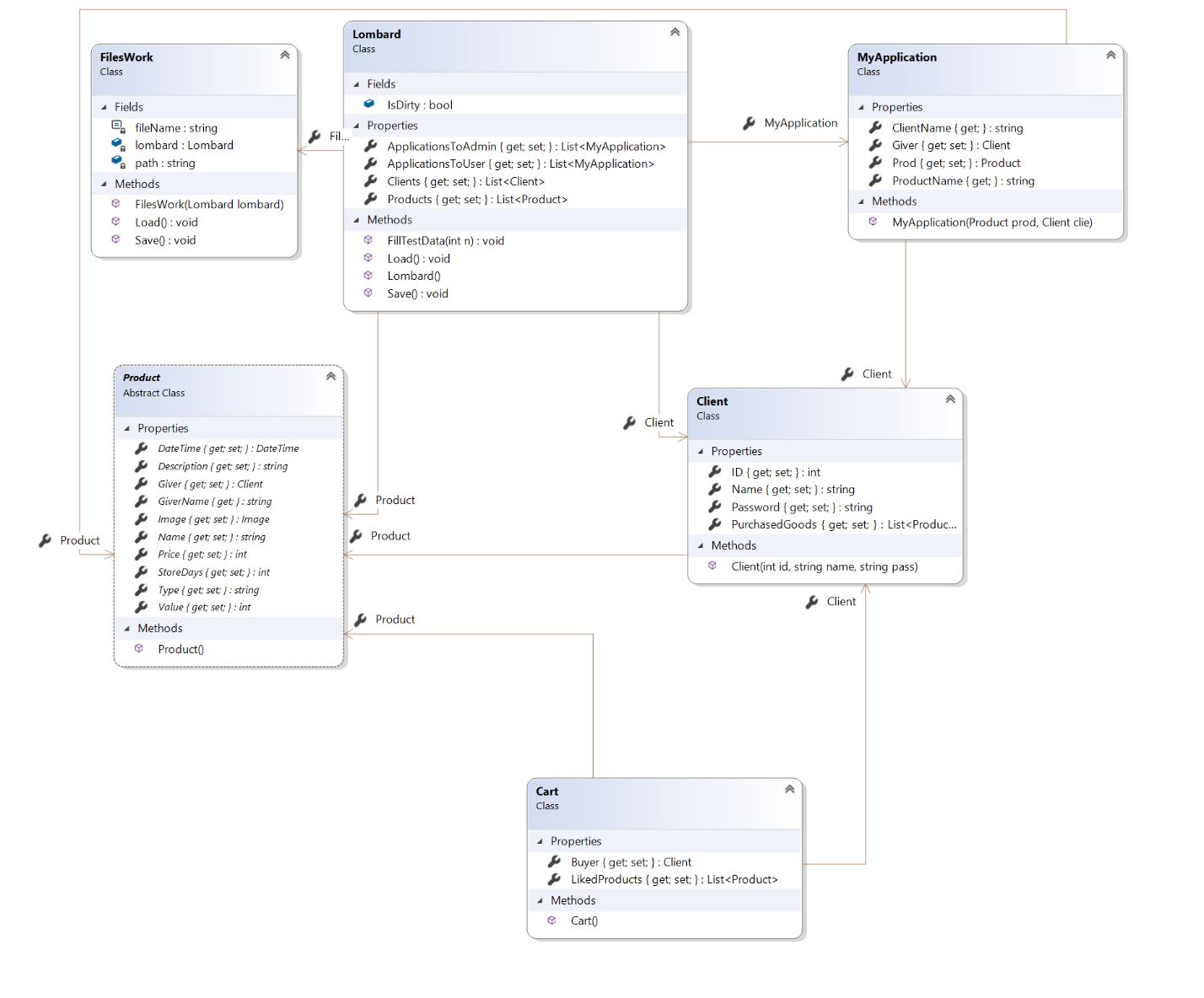


Рисунок 2.2 – Модель класів

З метою спрощення коду та запобігання його дублювання був використаний принцип поліморфізму у створенні класу «Квартира» та «Товар», що були успадковані від спільного класу «Предмет ». Їх відношення представлене на рисунку 2.3.



Рисунок 2.3 – Відношення наслідування між класами

* 1. Збереження та доступ до даних. Контроль версій

Дані переводяться з текстових в потік байтів за допомогою серіалізації з використанням вбудованої бібліотеки ВinaryFormatter. Захист даних в рамках даного проекту не розглядається.

Контроль версій проекту здійснюється за допомогою системи контроля версій GitHub.

**3 ІНСТРУКЦІЯ КОРИСТУВАЧА**

3.1 Запуск програми

Для запуску даної програми слід виконати такі дії:

1. Завантажити директорію з файлами курсового проекту.
2. Відкрити в директорії наступну папку: «AdminApp» , визначивши її вміст, знайти папку з назвою «bin».
3. Відкрити наступну папку, визначивши її вміст, знайти папку з назвою «Release».
4. Відкрити наступну папку, визначивши її вміст, знайти файли з назвами «AdminApp.exe», «Lombard\_Project.dll». Скопіювати їх до робочої директорії на своєму персональному комп’ютері.
5. Повернутися до директорії з файлами курсового проекту.
6. Відкрити в директорії наступну папку: «ClientApp» , визначивши її вміст, знайти папку з назвою «bin».
7. Відкрити наступну папку, визначивши її вміст, знайти папку з назвою «Release».
8. Відкрити наступну папку, визначивши її вміст, знайти файли з назвами «ClientApp.exe». Скопіювати їх до робочої директорії на своєму персональному комп’ютері.
9. У робочій директорії проекту запустити файл «ClientApp.exe» або «AdminApp.exe» для роботи з клієнтським або адміністративним додатком відповідно.
10. Програма запущена.
    1. Робота з програмою
       1. Додаток користувача

При запуску програми з’ являється вікно із входом до системи. Користувачу необхідно ввести свій логін та пароль, або натиснути на кнопку «Register», якщо він не є зареєстрованим у системі. У разі введення некоректних даних, поля підсвічуються червоним кольором. (Див. рис. 3.1 та 3.2)

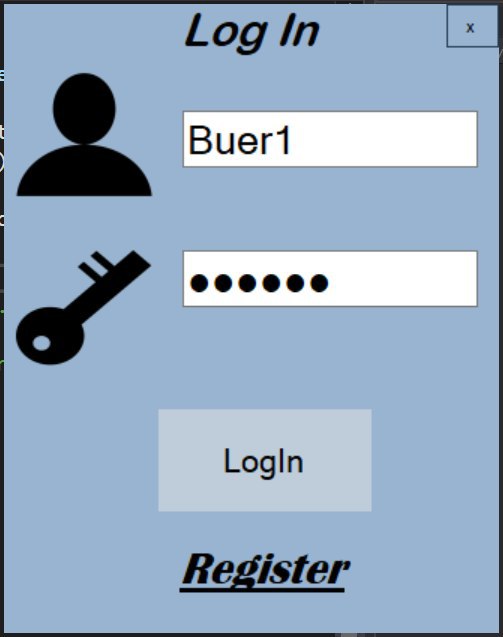


Рисунок 3.1 – Вікно входу

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 3.2 – Некоректні дані при вході

У разі реєстрації, відкривається форма для заповнення. Необхідно ввести логін, пароль та підтвердження паролю, номер паспорту. Вимоги вказані справа від поля введення. У разі недотримання вимог з’являється діалогове вікно із повідомленням про помилку. (Див. рис. 3.3)

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 3.3 – Вікно реєстрації

Після входу/реєстрації відкривається вікно із списком наявних у базі товарів та нерухомості. Вказується назва товару, його вартість, логін продавця, опис та фотографія. Користувач може виділяти товар, натиснувши на нього лівою кнопкою миші – тоді з’являється синє виділення. Виділений товар можна придбати, натиснувши на кнопку «Purchase». Можна відправити заявку на продаж продукту, натиснувши на «Send applications». (Див. рис. 3.4)

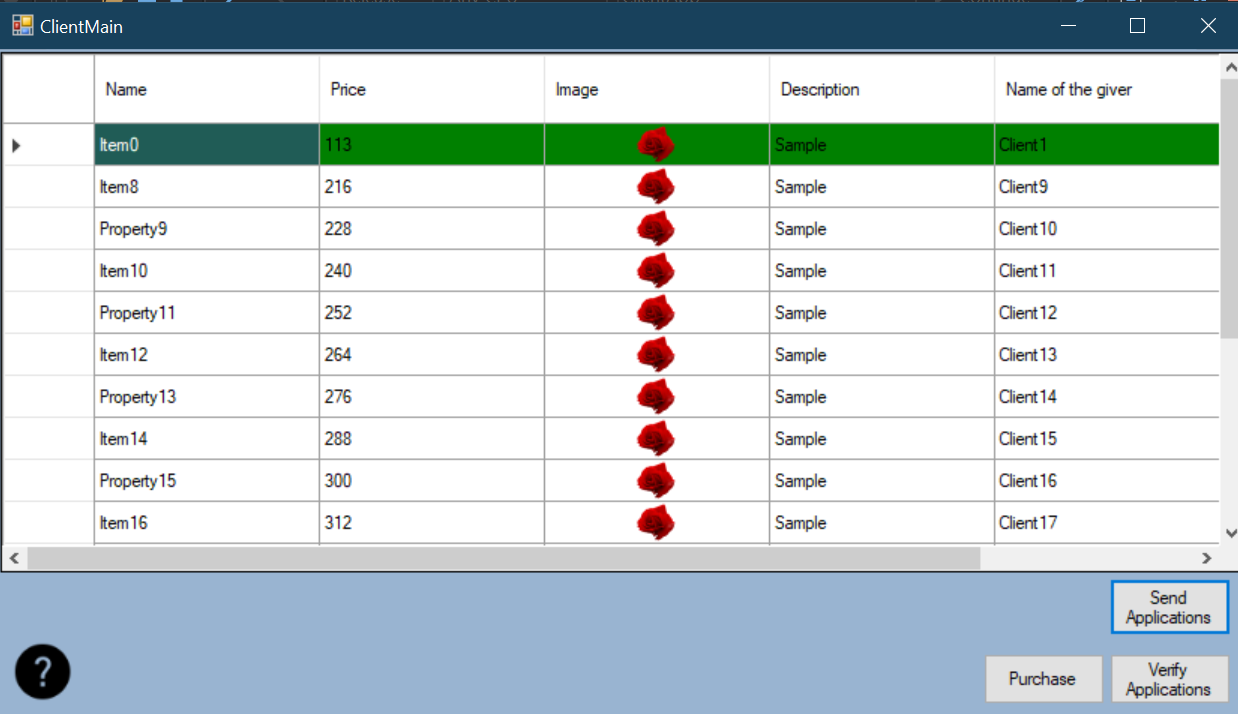


Рисунок 3.4 – Список товарів

При створенні заявки на продаж з’являється форма для заповнення. Необхідно вказати: назву товару, тип(нерухомість/предмет), оціночну вартість, опис товару та завантажити фотографію. Після натискання кнопки «Send application» заявка відсилається адміністратору на перевірку. (Див. рис. 3.5)

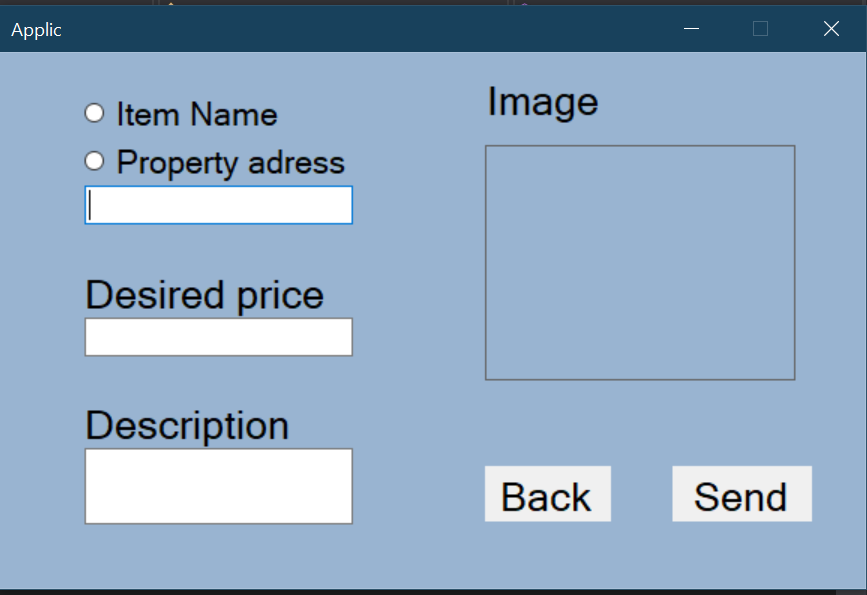


Рисунок 3.5 – Створення заявки

Натиснувши на «Verify applics» користувач отримує список своїх заявок, що були схвалені адміністратором. Якщо він згоден із внесеними змінами, він натискає «Accept» і заявка відображається в списку товарів. Якщо ні – натискає «Decline» і заявка видаляється з бази. (Див. рис. 3.6)

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 3.6 – Схвалення заявки

Користувачу доступна функція викупу товару, який він виставив на продаж протягом терміну зберігання. Заявки, доступні для викупу, виділені зеленим кольором та знаходяться у початку списку. (Див. рис. 3.7)

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 3.7 – Викуп своїх товарів

У разі покупки, необхідно виділити один чи декілька товарів та натиснути та кнопку «Purchase». Користувач потрапляє у корзину. Там він бачить список обраних товарів. Для покупки необхідно натиснути «Buy». У вкладці «Purchased» можна переглянути історію своїх придбань. (Див. рис. 3.8)

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 3.8 – Корзина

* + 1. Додаток для адміністратора

При запуску додатка адміністратора відкривається основне вікно із списком позицій, наявних у базі. Вказуються: назва товару, його вартість, логін продавця, опис та фотографія. Можна виділити найменування – вони виділяються синім кольором. (Див. рис. 3.9)

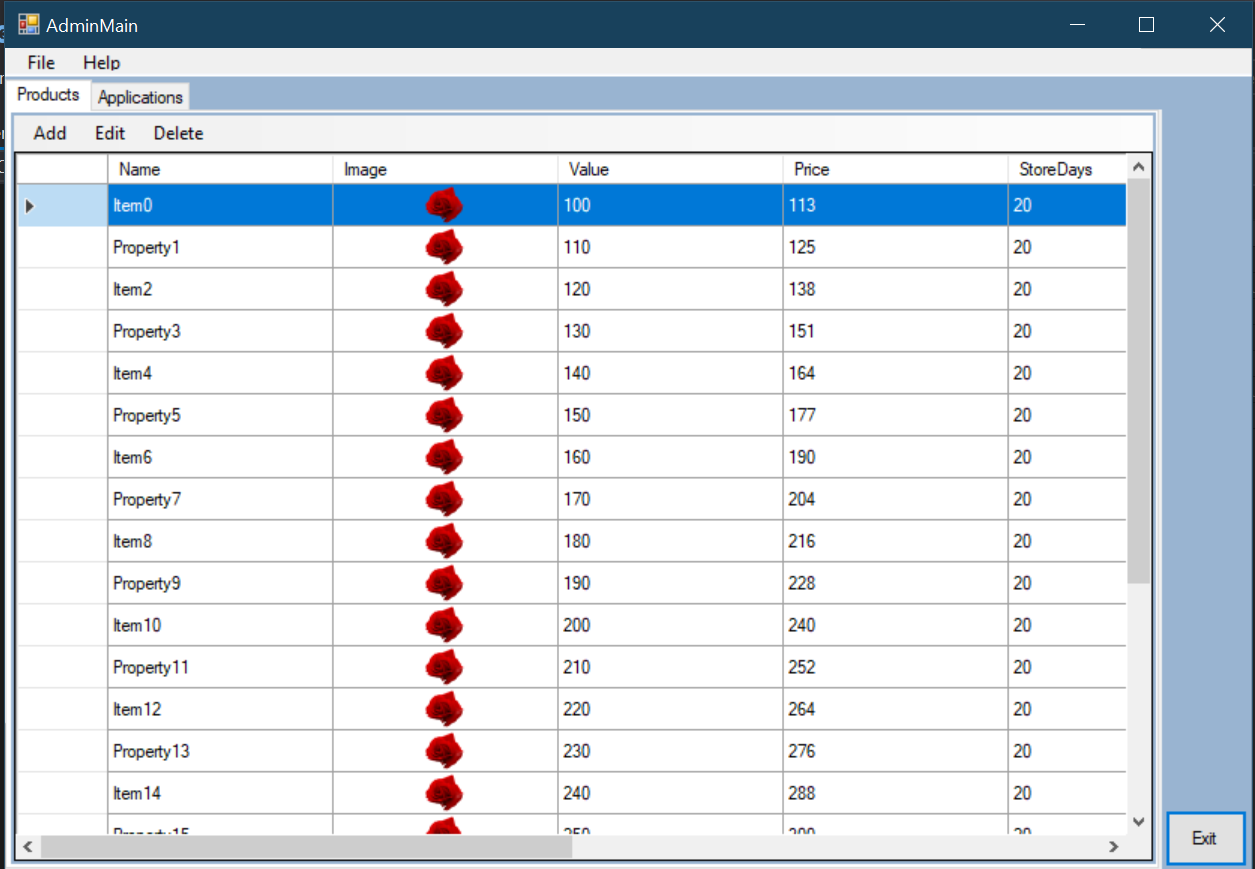


Рисунок 3.9 – Вікно адміністратора

Натиснувши на кнопку «Edit» адміністратор може відредагувати інформацію про виділений товар: назву, вартість, опис або фотографію. Відповідно натиснувши на «Delete» виділені товари будуть видалені з списку. (Див. рис. 3.10)

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 3.10 – Редагування товару

У вкладці «Applications» знаходяться заявки, що були відправлені користувачами на перевірку. Відкривши заявку, адміністратор може внести корективи, схвалити заявку (та відправити її на підтвердження клієнту) або відхилити її. (Див. рис. 3.11)

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 3.11 – Перевірка заявки

Адміністратор може також записати або вивести інформацію з файлу про стан товарів у базі, натиснувши File -> Save/Open. (Див. рис. 3.12)

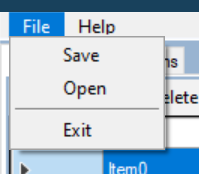


Рисунок 3.12– Запис до файлу

**ВИСНОВКИ**

За результатами виконання курсового проекту було створену програмне забезпечення «Ломбард», що зберігає та автоматично оновлює інформацію про доступні товари та нерухомість, надає змогу користувачам реєструватись у системі, робити покупки та дивитись історію своїх придбань. Також кожен користувач може заповнити заявку та, при умові схвалення її адміністратором, виставити свій товар на продаж. Відповідно адміністратор має змогу редагувати інформацію про існуючі позиції, видаляти їх із списку, схвалювати або відхиляти заявки.

Під час створення проекту було глибоко вивчено особливості та принципи об’єктно-орієнтованого програмування: взаємодію між класами, поняття наслідування та композиції, поліморфізм. Було розглянуто специфіку мови програмування С#. Також було детально вивчено особливості створення інтерфейсу користувача, взаємодію програмних класів із вікнами  Windows Forms. Усе це дало комплексне розуміння про процес створення програмного продукту від початку та до кінця: формулювання завдання, виділення чітких умов та критеріїв, візуалізація компонентів програми, розробка моделі класів та принципів взаємодії між ними.

Шляхами поліпшення запропонованої програми є розширення функціоналу, під’єднання до бази даних (для можливості перегляду списку продуктів з будь-якого ПК), та створення масштабного програмного забезпечення для мережі ломбардів, що було б здійснено на основі цього проекту.

**ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**

1. Бондарев В.М. Объектно-ориентированное программирование на С#: учебное пособие. Харьков: СМИТ, 2009г. 221 с.
2. Документація по C# від Microsoft. URL: https://docs.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 25.05.2020).
3. Документація по Windows Forms. URL: https://docs.microsoft.com/en-us/dotnet/framework/winforms/ (дата звернення: 25.05.2020).
4. Троелсен, Э, Джепикс, Ф. Язык программирования C# 7 и платформы .NET и .NET Core, 8-е изд. : Пер. с англ. СПб. : ООО “Диалектика”, 2018 — 1328 с. 978-5-6040723-1-8
5. Документація по .NET та C#. URL: https://metanit.com/sharp/ (дата звернення: 25.05.2020).
6. ДСТУ 3008:2015 Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. К.: ДП «УкрНДНЦ», 2016. 26 с.

**ДОДАТОК А**

Код програми

using Lombard\_Project.UserClasses;

using System.Collections.Generic;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

namespace Lombard\_Project.FilesWorkk

{

// Збереження та завантаження серіалізованих данних.

public class FilesWork

{

private Lombard lombard;

private string fileName = Path.Combine(Directory.GetParent(Directory.GetCurrentDirectory()).Parent.Parent.FullName, @"AdminApp/bin/Release/lombard.bin");

public FilesWork(Lombard lombard)

{

this.lombard = lombard;

}

// Збереження даних у файл.

public void Save()

{

using (Stream stream = File.Create(fileName))

{

var serializer = new BinaryFormatter();

serializer.Serialize(stream, lombard);

}

}

// Заввнтаження данних з файлу.

public void Load()

{

using (Stream stream = File.OpenRead(fileName))

{

var serializer = new BinaryFormatter();

Lombard lmd = (Lombard)serializer.Deserialize(stream);

Copy(lmd.Products, lombard.Products);

Copy(lmd.Clients, lombard.Clients);

Copy(lmd.ApplicationsToAdmin, lombard.ApplicationsToAdmin);

Copy(lmd.ApplicationsToUser, lombard.ApplicationsToUser);

}

void Copy<T>(List<T> from, List<T> to)

{

to.Clear();

to.AddRange(from);

}

}

}

}

using Lombard\_Project.FilesWorkk;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

namespace Lombard\_Project.UserClasses

{

[Serializable]

// Основний класс, що зберігає всі данні.

public class Lombard

{

// Коллекції для збереження предметів, користувачів та аплікацій.

public List<Product> Products { private set; get; }

public List<Client> Clients { private set; get; }

public List<MyApplication> ApplicationsToAdmin { private set; get; }

public List<MyApplication> ApplicationsToUser { private set; get; }

// Перевіряє чи змінені данні в файлу.

public bool IsDirty = false;

public Lombard()

{

Products = new List<Product>();

Clients = new List<Client>();

ApplicationsToAdmin = new List<MyApplication>();

ApplicationsToUser = new List<MyApplication>();

}

// Заповнює класс тестовими данними.

public void FillTestData(int n)

{

// Clients.

Clients.Clear();

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

{

Clients.Add(new Client(i + 10000, $"Client{i}", "12345"));

}

// Products.

Products.Clear();

var noImage = new Bitmap(Path.Combine(Directory.GetParent(Directory.GetCurrentDirectory()).Parent.Parent.FullName, @"ClientApp\Images\rose.png"));

for (int i = 0; i < n; i++)

{

if (i % 2 == 0)

{

Products.Add((Product)new Item($"Item{i}", 10 + i \* 10, Clients[i], "Sample", noImage, DateTime.Now - TimeSpan.FromDays(i + 16)) { Image = noImage, StoreDays = 20 });

}

else

{

Products.Add((Product)new Property($"Property{i}", 10 + i \* 10, Clients[i], "Sample", DateTime.Now - TimeSpan.FromDays(i + 16)) { Image = noImage, StoreDays = 20 });

}

}

// ApplicationsToAdmin.

ApplicationsToAdmin.Clear();

const int m = 5;

for (int i = 0; i < m; i++)

{

ApplicationsToAdmin.Add(new MyApplication(Products[0], Clients[0]));

}

// ApplicationsToUser.

ApplicationsToUser.Clear();

for (int i = 0; i < m; i++)

{

ApplicationsToUser.Add(new MyApplication(Products[0], Clients[0]));

}

}

// Зберегає данні.

public void Save()

{

new FilesWork(this).Save();

}

// Завантажує данні.

public void Load()

{

new FilesWork(this).Load();

for (int i = 0; i < Products.Count; i++)

{

int days = Convert.ToInt32((DateTime.Now - Products[i].DateTime).TotalDays);

if (days > Products[i].StoreDays)

Products[i].Price = Convert.ToInt32(1.2 \* Products[i].Value);

else

Products[i].Price = Convert.ToInt32((1 + (0.2 \* days / Products[i].StoreDays)) \* Products[i].Value);

}

}

}

}

using System;

using System.Drawing;

namespace Lombard\_Project.UserClasses

{

[Serializable]

// Абстрактний класс "Товару", який буде унаслідовано.

public abstract class Product

{

// Назва.

public abstract string Name { get; set; }

// Оціночна вартість.

public abstract int Value { get; set; }

// Дата та час здачі на зберігання.

public abstract DateTime DateTime { set; get; }

// Данні клієнта.

public abstract Client Giver { get; set; }

// Опис.

public abstract string Description { get; set; }

// Тип.

public abstract string Type { get; set; }

// Зображення.

public abstract Image Image { get; set; }

public abstract string GiverName { get; }

// Ціна на продажі.

public abstract int Price { get; set; }

// Термін зберігання.

public abstract int StoreDays { get; set; }

public Product()

{

}

}

}

using System;

using System.Drawing;

namespace Lombard\_Project.UserClasses

{

[Serializable]

// Класс "Квартира", унаслідує "Товар".

public class Property : Product

{

public override string Name { get; set; }

public override int Value { get; set; }

public override DateTime DateTime { set; get; }

public override Client Giver { get; set; }

public override string Description { get; set; }

public override string Type { get; set; }

public override Image Image { get; set; }

public override string GiverName { get => Giver.Name; }

public override int Price { set; get; }

public override int StoreDays { get; set; }

public Property(string name, int value, Client giver, string descr, DateTime? t = null)

{

Name = name;

Value = value;

Giver = giver;

Description = descr;

Type = "property";

if (t == null)

DateTime = DateTime.Now;

else

DateTime = (DateTime)t;

}

public Property() : base()

{

}

}

}

using System;

using System.Drawing;

namespace Lombard\_Project.UserClasses

{

[Serializable]

// Класс "Предмет", унаслідує "Товар".

public class Item : Product

{

public override string Name { get; set; }

public override int Value { get; set; }

public override DateTime DateTime { set; get; }

public override Client Giver { get; set; }

public override string Description { get; set; }

public override string Type { get; set; }

public override Image Image { get; set; }

public override string GiverName { get => Giver.Name; }

public override int Price { set; get; }

public override int StoreDays { get; set; }

public Item(string name, int value, Client giver, string descript, Image image, DateTime? t = null)

{

Name = name;

Value = value;

Giver = giver;

Description = descript;

Image = image;

Type = "item";

if (t == null)

DateTime = DateTime.Now;

else

DateTime = (DateTime)t;

}

public Item() : base()

{

}

}

}

using System;

using System.Collections.Generic;

namespace Lombard\_Project.UserClasses

{

[Serializable]

public class Client

{

// Паспорт клієнту.

public int ID { get; set; }

// Ім'я.

public string Name { get; set; }

// Пароль.

public string Password { get; set; }

// Колукція куплених товарів.

public List<Product> PurchasedGoods { set; get; }

public Client(int id, string name, string pass)

{

ID = id;

Name = name;

Password = pass;

PurchasedGoods = new List<Product>();

}

}

}

using System;

namespace Lombard\_Project.UserClasses

{

[Serializable]

// Класс аплікації.

public class MyApplication

{

// Який продукт здали.

public Product Prod { set; get; }

// Хто здав.

public Client Giver { set; get; }

// Повертає ім'я клієнта для коректного відображення.

public string ClientName

{

get { return Giver.Name; }

}

// Повертає назву товару для коректног відображення.

public string ProductName

{

get { return Prod.Name; }

}

public MyApplication(Product prod, Client clie)

{

Prod = prod;

Giver = clie;

}

}

}

using System;

namespace Lombard\_Project.UserClasses

{

[Serializable]

// Класс аплікації.

public class MyApplication

{

// Який продукт здали.

public Product Prod { set; get; }

// Хто здав.

public Client Giver { set; get; }

// Повертає ім'я клієнта для коректного відображення.

public string ClientName

{

get { return Giver.Name; }

}

// Повертає назву товару для коректног відображення.

public string ProductName

{

get { return Prod.Name; }

}

public MyApplication(Product prod, Client clie)

{

Prod = prod;

Giver = clie;

}

}

}