Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліни «Об’єктно-орієнтоване програмування»

на тему: «Біржа товарів»

Керівник, проф. Бондарєв В. М.

Изображение выглядит как зарисовка, Детское искусство, рисунок, Штриховая графика

Автоматически созданное описаниеСтудент гр. ПЗПІ-22-6 Головашенко І. В.

Комісія:

проф. Бондарєв В. М.,

ст. викл. Черепанова Ю. Ю.,

ст. викл. Ляпота В. М.

Харків 2023

Аркуш завдання та календарний план

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра: програмної інженерії

Рівень вищої освіти: перший (бакалаврський)

Дисципліна: Об’єктно-орієнтоване програмування

Спеціальність: 121 Інженерія програмного забезпечення

Освітня програма: Програмна інженерія

Курс: 1 . Група: ПЗПІ-22-6 . Семестр: 2 .

***ЗАВДАННЯ***

***На курсовий проект студента***

Головашенка Іллі Володимировича

1 Тема проекту: Біржа товарів

2 Термін здачі студентом закінченого проекту: «16» - червня – 2023 р.

3 Вихідні дані до проекту: Специфікація програми, методичні вказівки до виконання курсової роботи

4 Зміст розрахунково-пояснювальної записки: Вступ, опис вимог, проектування програми, інструкція користувача, висновки

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 13.02.2023 - 14.03.2023 р. |
| 2 | Формулювання вимог до програми | 02.05.2023 – 04.05.2023 р. |
| 3 | Розробка підсистеми зберігання та пошуку даних. | 23.05.2023 – 24.05.2023 р. |
| 4 | Розробка функцій регістрації користувача, пошуку та додавання товарів | 25.05.2023 – 29.05.2023 р. |
| 5 | Розробка функцій зберігання та завантаження даних | 29.05.2023 – 31.05.2023 р. |
| 6 | Тестування і доопрацювання розробленої програмної системи. | 01.06.2023 – 03.06.2023 р. |
| 7 | Оформлення пояснювальної записки, додатків, графічного матеріалу | 03.06.2023 – 07.06.2023 р. |
| 8 | Захист | 05.06.2023 – 16.06.2023 р. |

КАЛЕНДАРНИЙ ПЛАН

Изображение выглядит как зарисовка, Детское искусство, рисунок, Штриховая графика

Автоматически созданное описание

Студент

Керівник Головашенко І. В.

« 13 » лютого 2023 р.

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 60 с., 14 рис., 1 додаток, 5 джерел.

ПОКУПЕЦЬ, ПРОДАВЕЦЬ, МАГАЗИН, ЗВІТ, ООП, .NET, МОВА C#

Метою роботи є розробка програми «Біржа товарів», яка буде надавати зручну платформу, де покупці та продавці можуть здійснювати операції купівлі-продажу товарів. Застосунок має базу даних продавців та базу даних покупців, де зберігається інформація про особисті дані, додані товари та куплені або продані товари.

В результаті розробки отримана програма, що дозволяє додавати товари як продавцям (для продажу), так і покупцям (для покупки). Також кожна з сторін може знайти товари за різними параметрами, такими як назва товару, ціна, адреса самовивозу, форма оплати тощо. Після знаходження та обрання товару можна укласти договір купівлі-продажу. Всі укладені договори зберігаються в архівах як у продавця, так і у покупця.

В процесі розробки використано середовище Microsoft Visual Studio 2022, фреймворк Windows Forms, платформи .NET 7.0, мова програмування C#.

ЗМІСТ

Вступ……………………………………………………………………………….6

Розділ 1. Опис вимог до програми………………..……………………………...7

1.1 Опис сценаріїв………………….…………..…………………………7

1.2 Опис функцій…………...………………..…………………………..12

Розділ 2. Проектування програми……………...……………………………….21

2.1 Діаграма класів……………….……...……..………………………..21

2.2 Діаграма Use Case ………...……………..…………………………..22

2.3 Опис класів……………..……….…………..………………………..24

Розділ 3. Інструкція користувача………………………………………...……...31

3.1 Інструкція по розгортанню програми……..………………………..31

3.2 Інструкція користувача...………………..…………………………..32

Висновки…………………………………….…….……………………………..36

Перелік джерел посилання….................………………………………………..37

Додаток А………………………………………………………………………...38

ВСТУП

Метою курсового проекту є вивчення основних принципів ООП та набуття практичних навичок під час розробки власного програмного продукту. Основною метою проекту є створення програми “Біржа товарів”, що надасть покупцям та продавцям зручну платформу для укладення вигідних договорів купівлі-продажу.

Користувачі цієї програми зможуть зручно шукати товари за їх основними параметрами та додавати власні. Після успішного укладення договору купівлі-продажу користувачі зможуть переглянути деталі їх угоди у архіві. При відмові від послуг програма забезпечує повне видалення запиту, щоб уникнути непотрібного навантаження на систему.

Основна увага при розробці програми буде приділена створенню зручного та інтуїтивно зрозумілого інтерфейсу, що дозволить користувачам легко навігуватися по застосунку, додавати, шукати або переглядати товари з мінімальними зусиллями.

1. ОПИС ВИМОГ ДО ПРОГРАМИ
   1. Опис сценаріїв
      1. Сценарій 1. Реєстрація продавця/покупця

*Передумова*

Користувач відкрив програму та натиснув на посилання “Зареєструватись”.

*Основний сценарій*

1. Користувач заповнює всі необхідні поля у вікні реєстрації та натискає кнопку “Зареєструватись”.

2. Програма перевіряє введені дані, та створює обліковий запис покупця/продавця.

3. Програма сповіщає користувача про успішне завершення операції реєстрації.

*Додатковий сценарій*

1. Користувач заповнює необхідні поля у вікні реєстрації.

2. Програма перевіряє введені дані, поле “Логін”, що ввів користувач вже існує.

3. Програма сповіщає користувача про те, що такий Логін вже існує та пропонує придумати інший.

*Додатковий сценарій*

1. Користувач заповнює необхідні поля у вікні реєстрації.

2. Програма перевіряє коректність введених у поля даних, програма встановила, що дані не вірні.

3. Програма сповіщає користувача про неправильно введені дані та надає інструкцію, яким чином їх виправити.

* + 1. Сценарій 2. Вхід у обліковий запис

*Передумова*

Користувач відкривши програму опинився на вікні входу в обліковий запис.

*Основний сценарій*

1. Користувач заповнює всі необхідні поля у вікні входу та натискає кнопку “Увійти”.

2. Програма перевіряє коректність введених даних.

3. Після успішної перевірки програма перевіряє чи введений логін існує. Якщо так, то перевіряє пароль.

4. Пароль правильний, програма переводить користувача на основний екран застосунку.

*Додатковий сценарій*

1. Користувач заповнює необхідні поля у вікні входу.

2. Програма перевіряє коректність введених даних, вона встановила, що дані не коректні.

3. Програма сповіщає користувача про помилку в даних і про те, яким чином її виправити.

*Додатковий сценарій*

1. Користувач заповнює всі необхідні поля у вікні входу та натискає кнопку “Увійти”.

2. Програма перевіряє коректність введених даних.

3. Після успішної перевірки програма шукає обліковий запис у базі користувачів.

4. Якщо введений логін не був знайдений, програма пропонує користувачу перевірити введенні дані.

* + 1. Сценарій 3. Вихід з облікового запису

*Передумова*

Користувач пройшов аутентифікацію та знаходиться на основному вікні застосунку.

*Основний сценарій*

1. На головному вікні застосунку користувач може знайти кнопку “Вихід”.

2. При натисненні на кнопку “Вихід” програма запрошує користувача підтвердити операцію виходу.

3. При підтвердженні програма виходить з облікового запису користувача та переводить його на початковий екран.

*Додатковий сценарій*

1. При натисненні на кнопку “Вихід” програма запрошує користувача підтвердити операцію виходу.

2. Користувач не підтверджує операцію виходу.

3. Програма продовжує роботу на обліковому записі користувача.

* + 1. Сценарій 4. Додавання товару

*Передумова*

Користувач успішно пройшов аутентифікацію та натиснув у меню на кнопку “Додати товар”.

*Основний сценарій*

1. Користувач заповнює всі необхідні поля у вікні для додавання та натискає кнопку “Додати товар”.

2. Програма перевіряє коректність введених у поля даних, після успішної перевірки програма додає товар в базу даних товарів користувача.

3. Програма сповіщає про успішне завершення операції додавання товару.

*Додатковий сценарій*

1. Користувач заповнює необхідні поля у вікні для додавання товару.

2. Програма перевіряє коректність введених у поля даних, дані не коректні.

3. Програма сповіщає користувача про помилку в даних і про те, яким чином її виправити.

* + 1. Сценарій 5. Пошук товару

*Передумова*

Користувач успішно пройшов аутентифікацію та натиснув у меню кнопку “Знайти товар”.

*Основний сценарій*

1. Користувач заповнює хоча б одне поле за яким буде вестись пошук та натискає кнопку “Знайти товар”.

2. Програма перевіряє коректність введених у поля даних, після успішної перевірки програма шукає товар.

3. Після пошуку програма виводить список товарів, що задовольняє введеним даним.

*Додатковий сценарій*

1. Користувач заповнює хоча б одне поле за яким буде вестись пошук та натискає кнопку “Знайти товар”.

2. Програма перевіряє коректність введених у поля даних, після успішної перевірки програма шукає товар.

3. Якщо товарів не знайшлося програма сповіщає користувача “Таких товарів не знайдено”.

*Додатковий сценарій*

1. Користувач заповнює необхідні поля у вікні для пошуку товару.

2. Програма перевіряє коректність введених у поля даних, програма встановила, що дані не коректні.

3. Програма сповіщає користувача про неправильно введені дані та запрошує повторити ввід.

* + 1. Сценарій 6. Покупка товару

*Передумова*

Користувач успішно знайшов та обрав товар для покупки.

*Основний сценарій*

1. Користувач вводить кількість товару який хоче купити, та натискає кнопку “Купити”.

2. Програма перевіряє чи є в наявності такий обсяг товару.

3. Після успішної перевірки програма сповіщає про успішну покупку товару.

*Додатковий сценарій*

1. Користувач вводить кількість товару який хоче купити, та натискає кнопку “Купити”.

2. Програма перевіряє чи є в наявності такий обсяг товару.

3. Така кількість товару недоступна.

4. Програма сповіщає користувачу про перевищення кількості товару для покупки та запрошує повторити ввід.

* 1. Опис функцій
     1. Функція 1. Реєстрація продавця/покупця

Посилання на вікно реєстрації користувача знаходиться на початковому екрані. Вікно реєстрації містить такі поля:

1. два перемикачі – для вибору статусу користувача;
2. текстові поля:
   * 1. ім’я – справжнє ім’я користувача;
     2. прізвище – справжнє прізвище користувача;
     3. номер телефона – номер телефона користувача у форматі (+380-\*\*-\*\*\*-\*\*\*\*);
     4. логін – унікальна назва облікового запису користувача у програмі;
     5. пароль – певна послідовність символів, для підтвердження особи користувача.

На рисунку 1.1 наведено макет вікна реєстрації з усіма полями вводу.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 1.1 – Макет вікна реєстрації

В текстових полях а) та б) перша літера має бути верхнього регістру інші нижнього, в полях г) та д) регістр не важливий. В цих полях довжина рядка не перевищує 100 символів, для поля в) довжина рядка становить 16 символів.

Всі поля під час реєстрації мають бути заповнені.

Поля в) та г) мають містити унікальні значення для кожного користувача.

Внизу вікна знаходиться кнопка “Зареєструватись”. Натиснувши на неї програма перевіряє коректність усіх введених даних та реєструє користувача у базі даних.

Якщо дані не коректні, повідомлення про помилку буде з’являтись знизу від поля з помилкою.

* + 1. Функція 2. Вхід у обліковий запис

Вікно входу до облікового запису знаходиться на початковому екрані програми та містить такі текстові поля:

1. логін – унікальна назва облікового запису користувача у програмі;
2. пароль – певна послідовність символів, для підтвердження особи користувача.

На рисунку 1.2 наведено макет вікна входу з усіма полями вводу.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 1.2 – Макет вікна входу в обліковий запис

В текстових полях регістр не важливий, довжина рядка не перевищує 100 символів.

Всі поля під час аутентифікації мають бути заповнені.

Внизу вікна знаходиться кнопка “Увійти”. Натиснувши на неї програма перевіряє коректність усіх введених даних та виконує вхід у обліковий засіб.

Якщо дані не коректні, повідомлення про помилку буде з’являтись знизу від поля з помилкою.

* + 1. Функція 3. Вихід з облікового запису

Кнопка виходу з облікового запиcу розміщена на головному вікні програми.

При натисненні на неї програма запросить підтвердження операції виходу. Після підтвердження операції програма переведе користувача на початковий екран (див. рис. 1.2).

Изображение выглядит как текст, снимок экрана, Шрифт, Прямоугольник

Автоматически созданное описаниеНа рисунку 1.3 наведено макет головного вікна програми та кнопки виходу з облікового запису.

Рисунок 1.3 – Головне вікно застосунку та кнопка виходу з облікового запису

* + 1. Функція 4. Додавання товару

При натисненні кнопки “Додати товар” на головному вікні програми (див. рис. 1.3) користувач потрапить на вікно “Додання товару”. Це вікно містить такі текстові поля:

1. назва товару – фрагмент назви товару;
2. обсяг товару – ціле невід’ємне число;
3. ціна за одиницю – ціле невід’ємне число;
4. форма оплати – фрагмент назви способу оплати;
5. адреса для самовивозу – фрагмент адреси продавця/складу;
6. умова доставки – фрагмент умови доставки (напр. По всій Україні);
7. примітки – фрагмент нотатки для додаткового опису.

На рисунку 1.4 наведено макет вікна додавання товару з усіма полями вводу.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 1.4 – Макет вікна додання товару

В полях регістр а), д) та ж) регістр не важливий, максимальна довжина рядка – 100 символів, для полів б) та в) – 8 символів. Поля г) та е) є випадаючими списками. Обов’язковими для вводу є поля а) та б). Для інших полів відсутність даних у них означає, що вони не застосовується.

При натисненні на символ “<” програма запитає чи насправді користувач хоче припинити операцію. Після підтвердження його буде переведено на головне вікно (див. рис. 1.3).

Внизу знаходиться кнопка “Додати товар”. При натисненні на неї програма перевіряє коректність введених даних. Якщо ввід некоректний, то повідомлення про помилку буде з’являтись знизу від поля з помилкою. Інакше, товар буде додано до бази товарів.

* + 1. Функція 5. Пошук товару

При натисненні кнопки “Знайти товар” на головному вікні програми (див. рис. 1.3) користувач потрапить на вікно Пошуку товару. Це вікно містить такі текстові поля:

1. назва товару – фрагмент назви товару;
2. бажана ціна за одиницю – невід’ємне число;
3. форма оплати – фрагмент назви способу оплати;
4. адреса для самовивозу – фрагмент бажаної адреси продавця/складу;
5. умова доставки – фрагмент умови доставки (напр. По всій Україні);
6. примітки – фрагмент нотаток для додаткового опису.

На рисунку 1.5 наведено макет вікна пошуку товару з усіма полями вводу.

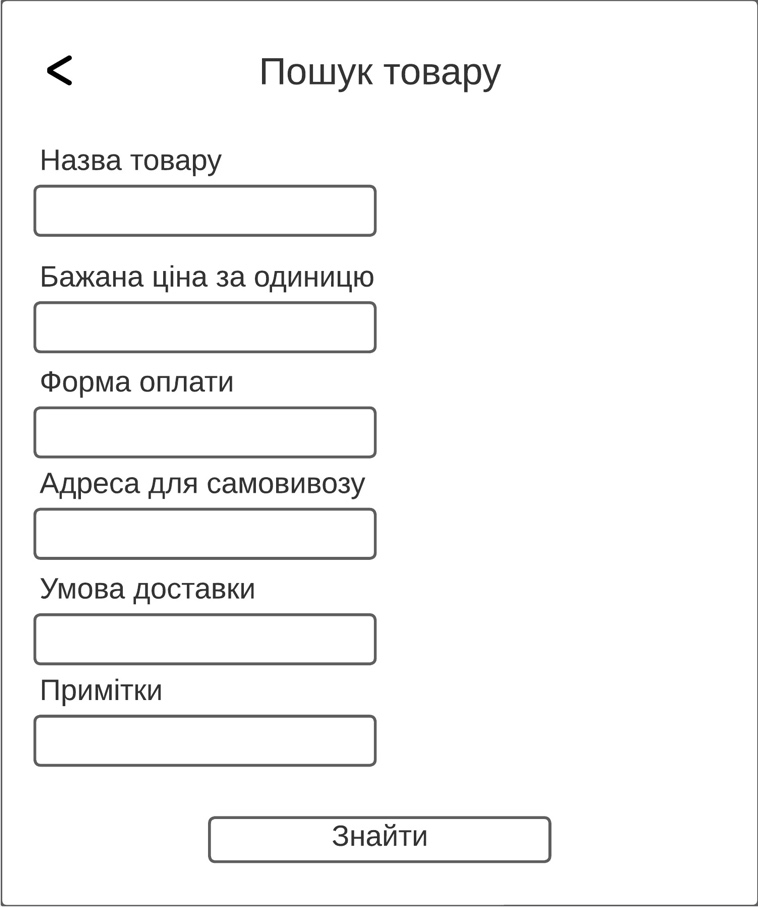


Рисунок 1.5 – Макет вікна пошуку товару

В полях а), г) та е) регістр не важливий, максимальна довжина рядка – 100 символів, для поля б) – 8 символів. Поля в) та д) є випадаючими списками. Обов’язковим для вводу є одне довільне поле. Для інших полів відсутність даних у них означає, що вони не застосовується.

При натисненні на символ “<” програма запитає чи насправді користувач хоче припинити операцію. Після підтвердження його буде переведено на головне вікно (див. рис. 1.3).

Внизу знаходиться кнопка “Знайти товар”. При натисненні на неї програма перевіряє коректність введених даних. Якщо ввід некоректний, то повідомлення про помилку буде з’являтись знизу від поля з помилкою.

Інакше, буде показано всі товари, які задовольняють критерії пошуку. Якщо таких товарів немає, програма виведе повідомлення “Таких товарів не знайдено”.

* + 1. Функція 6. Покупка товару

Після обрання товару, користувач потрапляє на вікно покупки товару. Це вікно містить двонаправлений лічильник:

1. Изображение выглядит как текст, снимок экрана, Шрифт, число

   Автоматически созданное описаниеКількість – кількість одиниць товару, що хоче купити користувач. Ціле, невід’ємне, відмінне від нуля число.

Рисунок 1.6 – Вікно покупки товару

Поле “Загальна вартість покупки” оновлюється щоразу, коли користувач збільшує або зменшує кількість одиниць товару.

При натисненні на символ “<” програма запитає чи насправді користувач хоче припинити операцію. Після підтвердження його буде переведено на вікно знайдених товарів.

При натисненні кнопки “Купити” програма перевіряє коректність введених даних. Після успішної перевірки кількість одиниць купленого товару віднімається від початкового обсягу цього товару.

Програма повідомляє користувача про успішне завершення операції покупки товару. Інакше, якщо ввід некоректний, то повідомлення про помилку буде з’являтись знизу від поля з помилкою[1].

1. ПРОЕКТУВАННЯ ПРОГРАМИ
   1. Діаграма класів

Изображение выглядит как текст, чек, диаграмма, документ

Автоматически созданное описаниеНа рисунку 2.1 зображено діаграму класів[2] для програми “Біржа товарів”.

Рисунок 2.1 – Діаграма класів для програми “Біржа товарів”

* 1. Діаграма Use Case

Изображение выглядит как текст, диаграмма, снимок экрана, круг

Автоматически созданное описаниеНа рисунку 2.2.1 зображено діаграму Use Case для застосунку “Біржа товарів”.

Рисунок 2.2 – Діаграма Use Case для застосунку “Біржа товарів”

* + 1. Специфікація елементів Use Case

Передумова: Користувач відкрив програму “Біржа товарів”.

Головний потік: Користувач проходить процес авторизації. Програма перевіряє його логін та пароль (E-1). Далі перед користувачем є вибір: “Знайти товар”, “Додати товар”, “Переглянути архів”, “Вийти”.

Якщо користувач обрав дію “Знайти товар”, то виконується під потік S-1: Пошук товару;

Якщо користувач обрав дію “Додати товар”, то виконується під потік S-2: Додання товару;

Якщо користувач обрав дію “Переглянути архів”, то виконується під потік S-3: Перегляд архіву;

Якщо користувач обрав дію “Вийти”, то виконується під потік S-4: Вихід з облікового запису.

Під потоки:

1. S-1: Пошук товару. Перед користувачем знаходиться вікно пошуку товару з полями для введення та кнопкою “Знайти товар” (див. рис. 1.5). Користувач заповнює необхідні поля та натискає кнопку (E-2). Програма виводить список товарів, які задовольняють параметрам пошуку. Користувач може обрати товар зі списку та купити-продати його (S-5);
2. S-2: Додання товару. Перед користувачем знаходиться вікно додання товару з полями для введення та кнопкою “Додати товар” (див. рис. 1.4). Користувач заповнює всі необхідні поля та натискає кнопку (E-2). Програма сповіщає користувача про успішне закінчення операції додання товару та переводить користувача на головне вікно програми;
3. S-3: Перегляд архіву. Перед користувачем знаходиться вікно зі списком укладених договорів купівлі-продажу. Якщо таких договорів ще не було укладено, програма сповіщає користувача про це;
4. S-4: Вихід з облікового запису. Програма пропонує користувачу підтвердити операцію виходу (E-3). Після підтвердження програма переводить користувача на вікно авторизації (див. рис. 1.2);
5. S-5: Купівля-продаж товару. Перед користувачем знаходиться вікно купівлі-продажу (див. рис. 1.6). Користувач вводить бажану кількість товару та натискає кнопку “Купити/Продати” (E-4). Програма сповіщає про успішне завершення операції та переводить користувача на головне вікно програми.

Альтернативні потоки:

1. E-1: Логін або пароль введений користувачем виявся некоректним. Програма сповіщає про помилку та пропонує повторити ввід або завершити елемент Use Case;
2. E-2: Користувач некоректно заповнив поля або не заповнив їх взагалі. Програма сповіщає про помилку та пропонує повторити ввід або завершити елемент Use Case;
3. E-3: Користувач не підтвердив вихід з облікового запису. Елемент Use Case завершується;
4. E-4: Недоступна кількість товару. Програма сповіщає про помилку та пропонує повторити ввід або завершити елемент Use Case.
   1. Опис класів
5. Клас User

Один з основних класів програми є абстрактним та публічним. Містить деякі поля, конструктор та властивість для представлення користувача. Основні атрибути класу:

* FullName – ім’я та прізвище користувача;
* Login – унікальна назва облікового запису користувача;
* Telephone – номер мобільного телефону користувача;
* ProductsPath – шлях до текстового файлу з продуктами, які додав користувач;
* Archive – шлях до текстового файлу з купленими-проданими користувачем продуктами.

Конструктор цього класу отримує масив даних, що містить інформацію для ініціалізації всіх полей.

Також клас має абстрактну властивість ClassName, що відповідає за статус користувача у програмі. Кожен клас, що успадковує User обов’язково має реалізувати цю властивість[2].

Клас User є абстрактним, що означає, що окремий екземпляр цього класу неможливо створити. Він служить базовим класом для класів, що наслідують його і надає загальну структуру для роботи з користувачами.

1. .1 Класи Salesman та Customer

Класи для представлення продавців та покупців у програмі. Успадковують клас User та реалізовують властивість СlassName.

Мають конструктор, який отримує масив даних, передає його базовому класу User та ініціалізує всі необхідні поля.

1. Клас Product

Другий з основних класів програми. Містить деякі поля, конструктор та методи для представлення об’єкту товару. Основні атрибути класу:

* Id – унікальний ідентифікатор товару;
* ProductName – назва товару;
* ProductAmount – кількість товару;
* ProductPrice – ціна за одиницю товару;
* PaymentForm – форма оплати товару;
* DeliveryCondition – умова доставки товару;
* Adress – адреса для самовивозу товару;
* Notes – додаткові примітки про товар;
* OwnerLogin – логін користувача, що додав цей товар.

Конструктор цього класу отримує масив даних, що містить інформацію для ініціалізації всіх полей.

Клас “Product” має статичний метод Search, який виконує пошук товарів за заданими параметрами. Він отримує список продуктів та словник який містить параметри пошуку, а повертає список знайдених продуктів. Метод проходиться по кожному елементу у списку та перевіряє чи відповідає він заданим параметрам. Якщо так, то продукт додається у список знайдених продуктів.

Також клас “Product” перевизначає метод ToString, який повертає рядок, що містить інформацію про продукт. Цей метод використовується для коректного відображення знайдених продуктів.

1. Клас Validators

Допоміжний статичний клас, який містить статичні методи для перевірки введених користувачем даних. Було вирішено використовувати методи цього класу під час події Control.Validating[3].

Основними методами класу є:

* ValidateField – статичний метод, який отримує текстове поле TextField вміст якого треба перевірити на відповідність вимогам, errorLabel типу Label (при некоректному вводі користувача в нього буде виводитись текст помилки), e типу CancelEventArgs (при неуспішній перевірці встановлюємо, що поле не пройшло валідацію) та делегат fieldValidationDelegate, за допомогою якого метод перевіряє значення текстового поля на коректність;
* AvailabilityCheck – статичний метод для перевірки наявності введеного у поле тексту у базі даних. Основним параметром, що перевіряє наявність тексту введеного користувачем в базі даних є делегат isAvailableDelegate. При неуспішній перевірці показує користувачу повідомлення помилки та пропонує повторити ввід.

Інші методи включають в себе перевірки різних полів, таких як ім’я, номер телефону, логін, пароль тощо. Вони перевіряють значення поля на відповідність необхідним вимогам використовуючи регулярні вирази[4] (наявність виключно латинських літер, правильний формат, перша велика буква). Повертають значення true, якщо текст поля успішно пройшов перевірку або встановлюють ErrorMessage (пояснення, чому виникла помилка) та значення false при неправильному вводі від користувача.

Також клас містить три типи делегатів FieldValidationDelegate, IsRightRegexDelegate та IsAvailableDelegate, для передачі функцій з заданими параметрами у вищезазначені методи.

1. Клас DataAccess

Допоміжний статичний клас, який містить статичні методи та константи для роботи з базою даних.

Константи:

* DataBasePath – відносний шлях до папки DataBase;
* CustomersData – шлях до бази даних покупців;
* SalesmenData – шлях до бази даних продавців;
* CustomerAddedProducts – шлях до папки в якій знаходяться текстові файли з бажаними покупцями товарами;
* SalesmenAddedProducts – шлях до папки в якій знаходяться текстові файли з доданими продавцями товарами;
* SaleArchive – шлях до папки з текстовими файлами архівів продавців;
* PurchasesArchive – шлях до папки з текстовими файлами архівів покупців.

Методи:

* WriteToDataBase – метод для запису даних у базу. Приймає шлях до файлу бази даних і дані, які треба записати та додає їх до файлу;
* GetItemFromDatabase – метод для отримання рядка даних з файлу за унікальним параметром. Приймає шлях до файлу і рядок з унікальним значенням. Зчитує кожний рядок текстового файлу та повертає перший рядок, який містить вказаний параметр;
* GetItemById – метод подібний до минулого, використовується для отримання даних про товар за унікальним ідентифікатором. Приймає шлях до папки з текстовими файлами та проходить по кожному файлу застосовуючи метод GetItemFromDatabase. Повертає перший знайдений рядок за заданим ідентифікатором;
* LoadData – метод, що використовується для завантаження всіх даних про товари з заданої папки. Приймає шлях до папки. Проходить по кожному рядку кожного файлу каталогу та спочатку створює об’єкти класу “Product” потім додає їх до списку, який у кінці повертає;
* LoadArchive – метод, що використовується для завантаження архіву товарів з бази даних. Приймає шлях до файлу архіву, зчитує файл по рядках та додає до списку, згодом повертає цей список;
* ChangeLineInDataBase – метод, що використовується для заміни або видалення рядка з бази даних. Приймає шлях до папки з текстовими файлами, старий рядок та новий рядок або рядок для видалення. Метод проходить по всім файлам у папці, створює тимчасовий файл у який записує вміст файла, коли доходить до рядка, що треба замінити, то пропускає його, якщо треба видалити інакше записує у тимчасовий файл новий рядок. У кінці тимчасовий файл стає на місце початкового.

1. Клас Utilities

Допоміжний статичний клас, який містить статичні методи для роботи з формами.

Методи:

* FiledsFilled – метод, який повертає кількість заповнених полів на формі. Приймає об’єкт типу Form та перебирає всі елементи керування на ньому. Якщо поле є текстовим та його текст не складається лише з пробілів або поле є комбінованим списком і має обраний елемент, то лічильник заповнених полів збільшується на 1. У кінці метод повертає значення лічильника;
* GetFilledFields – метод, який повертає словник, що містить пари “поле” і “значення” заповнених полів на формі. Приймає об’єкт типу Form та перебирає всі елементи керування на ньому. Якщо поле є текстовим та його текст не складається лише з пробілів або поле є комбінованим списком і має обраний елемент, то пара “поле” і “значення” додається до словника;
* ChangeForm – метод, який приховує поточну форму та відкриває нову форму типу T, яка має конструктор за замовчуванням;
* ConfirmOperation – метод, який викликає спливаюче вікно підтвердження з заданим текстом. Приймає об’єкти типу Form, User, string. Якщо користувач натисне кнопку “Yes”, то поточна форма приховається і відкриється форма типу MainForm;
* Getdata – метод, який приймає рядок з даними та кількість необхідних полей, а повертає масив, що заповнений необхідними значеннями для створення об’єктів типу “Customer”, “Salesman” або “Product” та довжина якого дорівнює кількості полей. Метод розбиває отриманий рядок використовуючи роздільник “; ” надалі з кожна пара отримана раніше знову розбивається за допомогою роздільника “: ”, значення, що йшло після роздільника “: ”, зберігається у результуючому масиві.

Це всі основні та допоміжні класи, що працюють у програмі “Біржа товарів”. Детальніше з класами та вихідним кодом програми, можете ознайомитись у додатку А.

1. ІНСТРУКЦІЯ КОРИСТУВАЧА
   1. Інструкція по розгортанню програми

Для розгортання програми на вашому персональному комп’ютері ви маєте виконати наступний перелік дій:

1. Переконайтеся, що ваша операційна система це Windows 7 або пізніша версія Windows;
2. Завантажте необхідні компоненти. Для коректної роботи програми переконайтеся, що на вашому комп’ютері встановлене необхідне середовище розробки (наприклад Visual Studio 2022) та необхідне розширення .NET desktop development (для Visual Studio) для підтримки проектів на Wіndows Forms та С#;
3. Розархівуйте завантажений з репозиторію GitHub zip-файл проекту та відкрийте його у вашому обраному середовищі розробки;
4. Скомпілюйте програму. Виконайте процес компіляції проекту, щоб отримувати виконуваний файл програми з розширенням (.exe);
5. Знайдіть програму. Після компіляції зайдіть у каталог проекту та перейдіть у новостворену папку з назвою “bin”. Пройшовши по папкам до кінця ви знайдете файл з назвою “Біржа товарів.exe”;
6. Відкрийте програму. Натиснувши швидко два рази на файл “Біржа товарів.exe” ви відкриєте застосунок.
   1. Інструкція по використанню програми
7. Завантажте та відкрийте програму “Біржа товарів” (див. пункт 3.1);
8. Зареєструйте свій обліковий запис. Для цього з вікна авторизації перейдіть на вікно реєстрації використовуючи посилання “Зареєструватись”. На вікні реєстрації заповніть всі поля та натисніть кнопку “Зареєструватись”; На рисунку 3.1 зображено коректно заповнені, всі поля форми;

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3.1 – Вікно реєстрації користувача. Коректне заповнення полей

1. Зареєструвавшись та увійшовши у свій акаунт ви опинитесь но головному вікні програми, яке зображено на рисунку 3.2;

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3.2 – Головне вікно застосунку

1. Додання товару. Натисніть на кнопку “Додати товар”. Перед вами відкриється вікно додання товару, зображене на рисунку 3.3. Для додання товару необхідно заповнити тільки два поля “Назва товару” та “Обсяг партії” інші поля заповнювати необов’язково. Після натиснення на кнопку “Додати товар” програма сповістить вас про успішне завершення операції;

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3.3 – Вікно додання товару. Коректний ввід даних у поля

1. Пошук товару. На головному вікні застосунку (див. рис. 3.2) натисніть кнопку “Знайти товар”. Відкриється вікно пошуку, зображене на рисунку 3.4. Для пошуку товарів необхідно заповнити хоча б одне поле. При натисненні кнопки “Знайти товар” програма продемонструє список знайдених товарів;

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3.4 – Вікно пошуку товарів

1. Купівля-продаж товару. Обравши товар зі списку ви можете купити його натиснувши на відповідну кнопку. Відкриється вікно купівлі-продажу, зображене на рисунку 3.5. Для купівлі-продажу вкажіть бажану кількість товару. При натисненні відповідної кнопки програма сповістить вас про успішне укладення угоди купівлі-продажу;

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3.5 – Вікно купівлі продажу

1. Перегляд архіву угод. Перейшовши на головне вікно (див. рис. 3.2) ви можете переглянути власний архів створених угод просто натиснувши відповідне посилання на вікні. Програма відкриє архів, що містить список укладених вами договорів купівлі-продажу. Вікно архіву зображено на рисунку 3.6.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 3.6 – Вікно архіву

ВИСНОВКИ

В результаті виконання курсового проекту на тему “Біржа товарів” було розроблено зручний застосунок для взаємодії покупців та продавців. Основними функціями програми є додавання, пошук та купівля товарів. Також вся інформація про користувачів та їх додані або куплені-продані товари зберігається в базі даних.

Під час роботи над курсовим проектом я навчився:

* працювати з середовищем розробки Visual Studio 2022 та розробив власний програмний продукт;
* працювати з файлами, формами, делегатами та регулярними виразами;
* будувати діаграми класів та Use Case діаграми.

Звісно, завжди є способи вдосконалення вже зробленого. Працюючи над проектом надалі можна додати наступний функціонал:

* розробити кошик для додання або видалення уподобаних товарів;
* можливість купити-продати одразу всі товари з кошику;
* можливість знову купити товар, який зберігається в архіві;
* впровадити систему поповнення балансу та систему створення електронних договорів;

Підсумовуючи, можна сказати, що розроблена програма повністю відповідає заявленим цілям та є зручним додатком для покупців та продавців.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методичні вказівки до виконання курсової роботи з дисципліни «Об’єктно-орієнтоване програмування» для студентів першого (бакалаврського) рівня вищої освіти усіх форм навчання спеціальності 121 «Інженерія програмного забезпечення», освітньо-професійна програма «Програмна інженерія» / Упоряд.: В.М. Бондарєв, Ю.Ю. Черепанова – Харків: ХНУРЕ, 2022. – 39 с.

2. Методичні вказівки до лабораторних робіт з дисципліни «Основи програмної інженерії» для студентів всіх форм навчання напряму – “Програмна інженерія” / Упоряд.: В.В. Голян, О.С. Ульянова. − Харків: ХНУРЕ, 2014. – 11 с.

3. Бондарєв В.М. Електронний учбовий посібник з дисципліни «Об’єктно-орієнтоване програмування» URL: http://tss.co.ua:5555/ (дата звернення: 05.05.2023)

4. .Net documentation. / Microsoft Learn. URL: <https://learn.microsoft.com/uk-ua/dotnet/api/system.windows.forms.control.validating> (дата звернення: 06.05.2023)

5. .Net documentation. / Microsoft Learn. URL: <https://learn.microsoft.com/uk-ua/dotnet/standard/base-types/regular-expressions> (дата звернення: 6.05.2023)

ДОДАТОК А

Вихідний код програми

using static Біржа\_товарів.Utilities.Validators;

using static Біржа\_товарів.Data.DataAccess;

using static Біржа\_товарів.Utilities.Utilities;

namespace Біржа\_товарів.Forms

{

// Клас, що реалізує форму для реєстрації користувача як продавця або покупця

public partial class RegistrationForm : Form

{

public RegistrationForm()

{

InitializeComponent();

}

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

{

Application.Exit();

}

private void ToAuthorizationFormLink\_LinkClicked(object sender,

LinkLabelLinkClickedEventArgs e)

{

ChangeForm<AuthorizationForm>(this);

}

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

{

bool IsSuccess = FieldsFilled(this) == 5;

if (!IsSuccess)

{

MainError.Text = "Всі поля мають бути заповнені";

}

else

{

MainError.Text = "";

string UserData = $"Ім'я: {NameField.Text}; " +

$"Прізвище: {SurnameField.Text}; Телефон: {PhoneField.Text}; " +

$"Логін: {LoginField.Text}; Пароль: {PasswordField.Text}; ";

if (SalesMan.Checked)

{

UserData += $"Товари: {SalesmenAddedProducts + LoginField.Text}Products.txt; " +

$"Архів: {SaleArchive + LoginField.Text}Archive.txt";

WriteToDataBase(SalesmenData, UserData);

}

else

{

UserData += $"Товари: {CustomerAddedProducts + LoginField.Text}Products.txt; " +

$"Архів: {PurchasesArchive + LoginField.Text}Archive.txt";

WriteToDataBase(CustomersData, UserData);

}

ChangeForm<AuthorizationForm>(this);

}

}

private void NameField\_Validating(object sender, CancelEventArgs e)

{

ValidateField(NameField, NameError, e, IsFieldValid);

}

private void NameField\_Validated(object sender, EventArgs e)

{

NameError.Text = "";

}

private void SurnameField\_Validating(object sender, CancelEventArgs e)

{

ValidateField(SurnameField, SurnameError, e, IsFieldValid);

}

private void SurnameField\_Validated(object sender, EventArgs e)

{

SurnameError.Text = "";

}

private void PhoneField\_Validating(object sender, CancelEventArgs e)

{

ValidateField(PhoneField, PhoneError, e, IsNumberValid);

AvailabilityCheck(PhoneField, PhoneError, "Телефон", e, IsAvailable);

}

private void PhoneField\_Validated(object sender, EventArgs e)

{

PhoneError.Text = "";

}

private void LoginField\_Validating(object sender, CancelEventArgs e)

{

ValidateField(LoginField, LoginError, e, IsLoginValid);

AvailabilityCheck(LoginField, LoginError, "Логін", e, IsAvailable);

}

private void LoginField\_Validated(object sender, EventArgs e)

{

LoginError.Text = "";

}

private void PasswordField\_Validating(object sender, CancelEventArgs e)

{

ValidateField(PasswordField, PasswordError, e, IsPasswordValid);

}

private void PasswordField\_Validated(object sender, EventArgs e)

{

PasswordError.Text = "";

}

}

}

using Біржа\_товарів.Forms;

using Біржа\_товарів.Models;

using static Біржа\_товарів.Utilities.Validators;

using static Біржа\_товарів.Data.DataAccess;

using static Біржа\_товарів.Utilities.Utilities;

namespace Біржа\_товарів

{

// Клас, що реалізує форму для входу в обліковий запис користувача

public partial class AuthorizationForm : Form

{

public AuthorizationForm()

{

InitializeComponent();

}

private void ToRegisterWindow\_LinkClicked(object sender,

LinkLabelLinkClickedEventArgs e)

{

ChangeForm<RegistrationForm>(this);

}

private void AuthorizationForm\_FormClosing(object sender,

FormClosingEventArgs e)

{

Application.Exit();

}

private void LoginField\_Validating(object sender, System.ComponentModel.CancelEventArgs e)

{

ValidateField(LoginField, LoginError, e, IsLoginValid);

}

private void LoginField\_Validated(object sender, EventArgs e)

{

LoginError.Text = "";

}

private void PasswordField\_Validating(object sender,

System.ComponentModel.CancelEventArgs e)

{

ValidateField(PasswordField, PasswordError, e, IsPasswordValid);

}

private void PasswordField\_Validated(object sender, EventArgs e)

{

PasswordError.Text = "";

}

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

{

bool IsSuccess = FieldsFilled(this) == 2;

if (!IsSuccess)

{

MainError.Text = "Всі поля мають бути заповнені";

}

else

{

MainError.Text = "";

string? user;

bool isSalesman = !IsAvailable(GetItemFromDatabase(SalesmenData,

$"Логін: {LoginField.Text};"));

bool isCustomer = !IsAvailable(GetItemFromDatabase(CustomersData,

$"Логін: {LoginField.Text};"));

if (isSalesman)

{

user = GetItemFromDatabase(SalesmenData, $"Логін: {LoginField.Text};");

if (user != null && user.Contains($"Пароль: {PasswordField.Text};"))

{

Salesman salesman = new Salesman(GetData(user, 7));

this.Hide();

MainForm mainForm = new MainForm(salesman);

mainForm.Show();

}

else

{

PasswordError.Text = "Неправильний пароль";

}

}

else if (isCustomer)

{

user = GetItemFromDatabase(CustomersData, $"Логін: {LoginField.Text};");

if (user != null && user.Contains($"Пароль: {PasswordField.Text};"))

{

Customer customer = new Customer(GetData(user, 7));

this.Hide();

MainForm mainForm = new MainForm(customer);

mainForm.Show();

}

else

{

PasswordError.Text = "Неправильний пароль";

}

}

else

{

LoginError.Text = "Перевірте правильність вашого логіну";

}

}

}

private bool IsAvailable(string? field)

{

return field == null;

}

}

}

using Біржа\_товарів.Models;

using static Біржа\_товарів.Models.Product;

using static Біржа\_товарів.Data.DataAccess;

using static Біржа\_товарів.Utilities.Utilities;

namespace Біржа\_товарів.Forms

{

// Клас, що реалізує головну форму з якої користувач може

// додати, знайти товари або переглянути куплені/продані товари

public partial class MainForm : Form

{

User user;

LinkedList<Product> CustomersWishes;

LinkedList<Product> SalesmenProducts;

public MainForm(User user)

{

InitializeComponent();

this.user = user;

SalesmenProducts = LoadData(SalesmenAddedProducts);

CustomersWishes = LoadData(CustomerAddedProducts);

}

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

{

this.Hide();

SearchProductForm searchProductForm = new SearchProductForm(user,

CustomersWishes, SalesmenProducts);

searchProductForm.Show();

}

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

{

this.Hide();

AddProductForm addProductForm = new AddProductForm(user);

addProductForm.Show();

}

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

{

Application.Exit();

}

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

{

RankText.Text = user.ClassName;

MeetingLabel.Text = $"Привіт, {user.FullName}!";

Archive.Text = user is Salesman ? "Продані товари" : "Куплені товари";

}

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

{

DialogResult result = MessageBox.Show("Ви точно хочете вийти?", "Підтвердіть вихід", MessageBoxButtons.YesNo);

if (result == DialogResult.Yes)

{

ChangeForm<AuthorizationForm>(this);

}

}

private void Archive\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

this.Hide();

ArchiveForm archiveForm = new ArchiveForm(user);

archiveForm.Show();

}

}

}

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Біржа\_товарів.Models;

using static Біржа\_товарів.Utilities.Validators;

using static Біржа\_товарів.Data.DataAccess;

using static Біржа\_товарів.Utilities.Utilities;

namespace Біржа\_товарів.Forms

{

// Клас, що реалізує форму для додавання товару продавцем або покупцем

public partial class AddProductForm : Form

{

User user;

static int Id = 1;

public AddProductForm(User user)

{

InitializeComponent();

this.user = user;

}

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

{

Application.Exit();

}

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

{

ConfirmOperation(this, user, "Ви підтверджуєте завершення цієї операції?");

}

private void ProductNameField\_Validating(object sender, CancelEventArgs e)

{

ValidateField(ProductNameField, ProductNameError, e, IsProductNameValid);

}

private void ProductNameField\_Validated(object sender, EventArgs e)

{

ProductNameError.Text = "";

}

private void ProductPriceField\_Validating(object sender, CancelEventArgs e)

{

ValidateField(ProductPriceField, ProductPriceError, e, IsProductPriceValid);

}

private void ProductPriceField\_Validated(object sender, EventArgs e)

{

ProductPriceError.Text = "";

}

private void AdressField\_Validating(object sender, CancelEventArgs e)

{

ValidateField(AdressField, AdressError, e, IsAdressValid);

}

private void AdressField\_Validated(object sender, EventArgs e)

{

AdressError.Text = "";

}

private void ProductAmountField\_Validating(object sender, CancelEventArgs e)

{

ValidateField(ProductAmountField, AmountError, e, IsProductPriceValid);

}

private void ProductAmountField\_Validated(object sender, EventArgs e)

{

AmountError.Text = "";

}

private void NotesField\_Validating(object sender, CancelEventArgs e)

{

ValidateField(NotesField, NotesError, e, IsProductNameValid);

}

private void NotesField\_Validated(object sender, EventArgs e)

{

NotesError.Text = "";

}

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

{

if (!(string.IsNullOrWhiteSpace(ProductNameField.Text)

|| string.IsNullOrWhiteSpace(ProductAmountField.Text)))

{

string ProductData = $"Id: {Id}; Назва: {ProductNameField.Text}; " +

$"Ціна: {ProductPriceField.Text}; Кількість: {ProductAmountField.Text}; " +

$"Адреса: {AdressField.Text}; Оплата: {PaymentFormBox.Text}; " +

$"Доставка: {DeliveryConditionBox.Text}; Замітки: {NotesField.Text}; " +

$"Власник: {user.Login}";

WriteToDataBase(user.ProductsPath, ProductData);

MessageBox.Show("Ви успішно додали товар!");

this.Hide();

MainForm mainForm = new MainForm(user);

mainForm.Show();

}

else

{

MainError.Text = "Поля для назви та кількісті товарів мають бути заповнені";

}

}

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

{

if (GetItemById(CustomerAddedProducts, $"Id: {Id};") != null

|| GetItemById(SalesmenAddedProducts, $"Id: {Id};") != null)

{

Id++;

AddProductForm\_Load(sender, e);

}

}

}

}

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Біржа\_товарів.Models;

using static Біржа\_товарів.Utilities.Validators;

using static Біржа\_товарів.Utilities.Utilities;

using static Біржа\_товарів.Models.Product;

using System.Security.AccessControl;

namespace Біржа\_товарів.Forms

{

// Клас, що реалізує форму для пошуку товару продавцем або покупцем

public partial class SearchProductForm : Form

{

User user;

internal LinkedList<Product> CustomersWishes;

internal LinkedList<Product> SalesmenProducts;

public SearchProductForm(User user, LinkedList<Product> wishes,

LinkedList<Product> products)

{

InitializeComponent();

this.user = user;

CustomersWishes = wishes;

SalesmenProducts = products;

}

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

{

ConfirmOperation(this, user,

"Ви підтверджуєте завершення цієї операції?");

}

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

{

Application.Exit();

}

private void ProductNameField\_Validating(object sender, CancelEventArgs e)

{

ValidateField(ProductNameField, ProductNameError, e, IsProductNameValid);

}

private void ProductNameField\_Validated(object sender, EventArgs e)

{

ProductNameError.Text = "";

}

private void ProductPriceField\_Validating(object sender, CancelEventArgs e)

{

ValidateField(ProductPriceField,

ProductPriceError, e, IsProductPriceValid);

}

private void ProductPriceField\_Validated(object sender, EventArgs e)

{

ProductPriceError.Text = "";

}

private void NotesField\_Validating(object sender, CancelEventArgs e)

{

ValidateField(NotesField, NotesError, e, IsProductNameValid);

}

private void NotesField\_Validated(object sender, EventArgs e)

{

NotesError.Text = "";

}

private void AdressField\_Validating(object sender, CancelEventArgs e)

{

ValidateField(AdressField, AdressError, e, IsAdressValid);

}

private void AdressField\_Validated(object sender, EventArgs e)

{

AdressError.Text = "";

}

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

{

bool IsSuccess = FieldsFilled(this) >= 1;

if (IsSuccess)

{

MainError.Text = "";

Dictionary<string, string> searchParameters = GetFilledFields(this);

List<Product> products;

products = user is Salesman ? Search(CustomersWishes,

searchParameters) : Search(SalesmenProducts, searchParameters);

this.Hide();

ProductSelectionForm selectionForm = new ProductSelectionForm(user, products);

selectionForm.Show();

}

else

{

MainError.Text = "Для пошуку має бути заповнено хоча б одне поле";

}

}

}

}

using Біржа\_товарів.Models;

using static Біржа\_товарів.Utilities.Utilities;

namespace Біржа\_товарів.Forms

{

// Клас, що реалізує форму для перегляду знайдених товарів та обрання бажаного товару для покупки

public partial class ProductSelectionForm : Form

{

User user;

List<Product> foundProducts;

public ProductSelectionForm(User user, List<Product> products)

{

InitializeComponent();

this.user = user;

foundProducts = products;

}

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

{

PurchaseButton.Text = user is Salesman ? "Продати" : "Купити";

if (foundProducts.Count == 0)

{

foundProductList.Items.Add("Таких товарів не знайдено");

foundProductList.Enabled = false;

}

else

{

foundProductList.DataSource = foundProducts;

}

}

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

{

ConfirmOperation(this, user, "Ви підтверджуєте завершення цієї операції?");

}

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

{

if (foundProductList.SelectedItem != null)

{

Product selectedProduct = (Product)foundProductList.SelectedItem;

this.Hide();

PurchaseForm purchaseForm = new PurchaseForm(user,

selectedProduct, foundProducts);

purchaseForm.Show();

}

else

{

PurchaseError.Text = "Оберіть товар для покупки";

}

}

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

{

Application.Exit();

}

}

}

using Біржа\_товарів.Models;

using System.Text.RegularExpressions;

using static Біржа\_товарів.Data.DataAccess;

namespace Біржа\_товарів.Forms

{

// Клас, що реалізує форму для купівлі-продажу товару

public partial class PurchaseForm : Form

{

User user;

List<Product> foundedProducts;

Product SelectedProduct;

public PurchaseForm(User user,

Product selectedProduct, List<Product> foundedProducts)

{

InitializeComponent();

this.user = user;

SelectedProduct = selectedProduct;

this.foundedProducts = foundedProducts;

}

private void PurchaseForm\_FormClosing(object sender,

FormClosingEventArgs e)

{

Application.Exit();

}

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

{

PurchaseFormLabel.Text = user is Salesman ? "Продаж товару" : "Покупка товару";

ProductNameField.Text = SelectedProduct.ProductName;

TotalPrice.Text = (Amount.Value \* SelectedProduct.ProductPrice).ToString() + " грн";

BuyButton.Text = user is Salesman ? "Продати" : "Купити";

}

private void Amount\_ValueChanged(object sender, EventArgs e)

{

TotalPrice.Text = (Amount.Value \* SelectedProduct.ProductPrice).ToString() + " грн";

}

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

{

DialogResult result = MessageBox.Show("Ви точно хочете припинити операцію?",

"Підтвердіть припинення операції", MessageBoxButtons.YesNo);

if (result == DialogResult.Yes)

{

this.Hide();

ProductSelectionForm selectionForm = new ProductSelectionForm(user,

foundedProducts);

selectionForm.Show();

}

}

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

{

string PurchaseData = $"Назва: {ProductNameField.Text}; " +

$"Кількість: {Amount.Value}; Ціна: {TotalPrice.Text}; " +

$"Дата та час укладення угоди: {DateTime.Now.ToString("dd/MM/yy HH:mm:ss")}";

string? line = GetItemById(user is Salesman ?

CustomerAddedProducts : SalesmenAddedProducts, $"Id: {SelectedProduct.Id};");

if (Amount.Value > 0 && Amount.Value <= SelectedProduct.ProductAmount)

{

WriteToDataBase(user.Archive, PurchaseData);

string? owner = GetItemFromDatabase(user is Salesman ?

CustomersData : SalesmenData, $"Логін: {SelectedProduct.OwnerLogin};");

string ownerArchivePath = owner is not null ? owner.Split("Архів: ")[1] : "";

WriteToDataBase(ownerArchivePath, PurchaseData);

if (Amount.Value == SelectedProduct.ProductAmount)

{

ChangeLineInDataBase(user is Salesman ?

CustomerAddedProducts : SalesmenAddedProducts, null, null, line);

}

else

{

string newValue = $"Кількість: {SelectedProduct.ProductAmount - Amount.Value}";

string newLine = line is not null ? Regex.Replace(line, @"Кількість:\s+\d+", newValue) : "";

ChangeLineInDataBase(user is Salesman ?

CustomerAddedProducts : SalesmenAddedProducts, line, newLine, null);

}

this.Hide();

MainForm mainForm = new MainForm(user);

mainForm.Show();

}

else

{

AmountError.Text = "Недоступна кількість";

}

}

}

}

using Біржа\_товарів.Models;

using static Біржа\_товарів.Data.DataAccess;

namespace Біржа\_товарів.Forms

{

// Клас, що реалізує форму для перегляду історії купівель/продаж

public partial class ArchiveForm : Form

{

User user;

public ArchiveForm(User user)

{

InitializeComponent();

this.user = user;

}

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

{

ArchiveLabel.Text = user is Salesman ? "Продані товари" : "Куплені товари";

ArchiveList.HorizontalScrollbar = true;

string ArchivePath = user is Salesman ? SaleArchive : PurchasesArchive;

ArchivePath += user.Login + "Archive.txt";

ArchiveList.DataSource = LoadArchive(ArchivePath);

if (ArchiveList.Items.Count == 0)

{

ArchiveList.Items.Add(user is Salesman ?

"Ви не продали жодного товару" : "Ви не купили жодного товару");

}

}

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

{

this.Hide();

MainForm mainForm = new MainForm(user);

mainForm.Show();

}

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

{

Application.Exit();

}

}

}

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

using static Біржа\_товарів.Data.DataAccess;

namespace Біржа\_товарів.Utilities

{

// Статичний клас, що реалізує методи для перевірки введених користувачем даних

public static class Validators

{

// Основний метод для перевірки даних

// при успішній перевірці закінчує процес валідації

// при невдалій показує помилку користувача при введені

// запрошує повторити ввід

public static void ValidateField(TextBox textField, Label errorLabel,

CancelEventArgs e, FieldValidationDelegate fieldValidationDelegate)

{

string errorMessage;

if (!fieldValidationDelegate(textField.Text, out errorMessage))

{

e.Cancel = true;

textField.Select(0, textField.Text.Length);

errorLabel.Text = errorMessage;

}

}

// Основний метод для перевірки наявності введеного тексту в базі данних

public static void AvailabilityCheck(TextBox textField, Label errorLabel,

string FieldName, CancelEventArgs e, IsAvailableDelegate isAvailableDelegate)

{

string errorMessage;

if (!isAvailableDelegate(FieldName, textField.Text, out errorMessage))

{

e.Cancel = true;

textField.Select(0, textField.Text.Length);

errorLabel.Text = errorMessage;

}

}

// Метод для перевірки імені та прізвища користувача

public static bool IsFieldValid(string Name, out string ErrorMessage)

{

return HasLetter(Name, out ErrorMessage)

&& IsRightRegex(Name, out ErrorMessage,

new Regex(@"^[А-ЩЬЮЯЇІЄҐа-щьюяїієґ]+$"),

"Поле має містити виключно українські літери")

&& IsFirstLetterCapital(Name, out ErrorMessage);

}

// Метод для перевірки правильного формату номера

public static bool IsNumberValid(string Name, out string ErrorMessage)

{

return HasLetter(Name, out ErrorMessage)

&& IsRightRegex(Name, out ErrorMessage,

new Regex(@"^\+380-\d{2}-\d{3}-\d{4}$"), "Неправильний формат номера");

}

// Метод для перевірки логіну

public static bool IsLoginValid(string Name, out string ErrorMessage)

{

return HasLetter(Name, out ErrorMessage)

&& IsRightRegex(Name, out ErrorMessage,

new Regex("^[a-zA-Z0-9]+$"),

"Поле має має містити латинські літери або цифри");

}

// Метод для перевірки пароля

public static bool IsPasswordValid(string Name, out string ErrorMessage)

{

return HasLetter(Name, out ErrorMessage)

&& IsRightRegex(Name, out ErrorMessage,

new Regex("^[a-zA-Z0-9]+$"),

"Поле має має містити латинські літери або цифри")

&& IsEnoughLetter(Name, out ErrorMessage);

}

// Метод для перевірки імені товара

public static bool IsProductNameValid(string Name, out string ErrorMessage)

{

return ProductFieldsValidating(Name, out ErrorMessage,

new Regex(@"^[\p{L}\d\s,]+$"), "Поле має містити літери або цифри", IsRightRegex);

}

// Метод для перевірки вартості та кількості товарів

public static bool IsProductPriceValid(string Name, out string ErrorMessage)

{

return ProductFieldsValidating(Name, out ErrorMessage,

new Regex(@"^(?!0+$)\d+$"), "Поле має містити числа більші за нуль", IsRightRegex);

}

// Метод для перевірки коректного вводу адреси

public static bool IsAdressValid(string Name, out string ErrorMessage)

{

return ProductFieldsValidating(Name, out ErrorMessage,

new Regex(@"^[А-ЩЬЮЯЇІЄҐа-щьюяїієґ]+,\s[А-ЩЬЮЯЇІЄҐа-щьюяїієґ\s]+,\s[А-ЩЬЮЯЇІЄҐа-щьюяїієґ\s]+\s\d+$"),

"Введіть аресу у форматі: Країна, Місто, Вулиця Номер", IsRightRegex);

}

// Метод для перевірки наявності тексту в базі даних

public static bool IsAvailable(string FieldName, string Name, out string ErrorMessage)

{

if ((GetItemFromDatabase(CustomersData, $"{FieldName}: {Name};") == null)

&& (GetItemFromDatabase(SalesmenData, $"{FieldName}: {Name};")) == null)

{

ErrorMessage = "";

return true;

}

ErrorMessage = $"Такий {FieldName} вже зареєстровано";

return false;

}

private static bool HasLetter(string Name, out string ErrorMessage)

{

if (Name.Length == 0)

{

ErrorMessage = "Поле має бути заповнено";

return false;

}

ErrorMessage = "";

return true;

}

private static bool IsFirstLetterCapital(string Name, out string ErrorMessage)

{

char firstLetter = Name[0];

if (!(char.IsUpper(firstLetter) && Name.Substring(1).All(char.IsLower)))

{

ErrorMessage = "Поле має починатись з великої літери";

return false;

}

ErrorMessage = "";

return true;

}

private static bool IsRightRegex(string Name,

out string ErrorMessage, Regex regex, string ErrorString)

{

if (!regex.IsMatch(Name))

{

ErrorMessage = ErrorString;

return false;

}

ErrorMessage = "";

return true;

}

private static bool IsEnoughLetter(string Name, out string ErrorMessage)

{

if (Name.Length < 8)

{

ErrorMessage = "Поле має бути довше 8-ми символів";

return false;

}

ErrorMessage = "";

return true;

}

private static bool ProductFieldsValidating(string Name, out string ErrorMessage,

Regex regex, string errorString, IsRightRegexDelegate isRightRegexDelegate)

{

if (string.IsNullOrWhiteSpace(Name))

{

ErrorMessage = "";

return true;

}

return isRightRegexDelegate(Name, out ErrorMessage, regex, errorString);

}

public delegate bool FieldValidationDelegate(string name,

out string errorMessage);

public delegate bool IsRightRegexDelegate(string name,

out string errorMessage, Regex regex, string errorString);

public delegate bool IsAvailableDelegate(string FieldName,

string name, out string errorMessage);

}

}

using Біржа\_товарів.Forms;

using Біржа\_товарів.Models;

namespace Біржа\_товарів.Utilities

{

// Статичний клас для допоміжних методів

internal static class Utilities

{

// Метод який повертає кількість заповнених полів

public static int FieldsFilled(Form form)

{

int counter = 0;

foreach (Control control in form.Controls)

{

if (control is TextBox textBox)

{

if (!string.IsNullOrWhiteSpace(textBox.Text))

{

counter++;

}

}

else if (control is ComboBox comboBox)

{

if (comboBox.SelectedItem != null)

{

counter++;

}

}

}

return counter;

}

// Метод який повертає словник<поле, значення> заповнених полів

public static Dictionary<string, string> GetFilledFields(Form form)

{

Dictionary<string, string> filledFields = new Dictionary<string, string>();

foreach (Control control in form.Controls)

{

if (control is TextBox textBox)

{

if (!string.IsNullOrWhiteSpace(textBox.Text))

{

filledFields.Add(textBox.Name.Replace("Field", ""), textBox.Text);

}

}

else if (control is ComboBox comboBox)

{

if (comboBox.SelectedItem != null)

{

filledFields.Add(comboBox.Name.Replace("Box", ""), comboBox.Text);

}

}

}

return filledFields;

}

// Метод для заміни форми конструктор якої не потребує параметрів

public static void ChangeForm<T>(Form thisForm) where T : Form, new()

{

thisForm.Hide();

T newForm = new T();

newForm.Show();

}

// Метод для виклику спливаючого вікна підтвердження

public static void ConfirmOperation(Form form, User user, string text)

{

DialogResult result = MessageBox.Show(text,

"Підтвердіть припинення операції", MessageBoxButtons.YesNo);

if (result == DialogResult.Yes)

{

form.Hide();

MainForm mainForm = new MainForm(user);

mainForm.Show();

}

}

// Метод, що повертає масив значень полів класів User або Product

public static string[] GetData(string Data, int fieldsCount)

{

string[] result = new string[fieldsCount];

string[] DataPairs = Data.Split("; ");

for (int i = 0; i < DataPairs.Length; i++)

{

result[i] = DataPairs[i].Split(": ")[1];

}

return result;

}

}

}

using Біржа\_товарів.Models;

using static Біржа\_товарів.Utilities.Validators;

using static Біржа\_товарів.Utilities.Utilities;

using System.Web;

namespace Біржа\_товарів.Data

{

// Статичний клас для роботи з базою даних

internal static class DataAccess

{

public const string DataBasePath = @"..\..\..\DataBase\";

public const string CustomersData = DataBasePath + "Customers.txt";

public const string SalesmenData = DataBasePath + "Salesmen.txt";

public const string SalesmenAddedProducts = DataBasePath + @"SalesmenAddedProducts\";

public const string CustomerAddedProducts = DataBasePath + @"CustomersAddedProducts\";

public const string SaleArchive = DataBasePath + @"SaleArchive\";

public const string PurchasesArchive = DataBasePath + @"PurchasesArchive\";

// Метод для запису даних у базу даних

public static void WriteToDataBase(string Path, string Data)

{

using (StreamWriter sw = new StreamWriter(Path, true))

{

sw.WriteLine(Data);

}

}

// Метод для отримання рядка данних з файлу за унікальним параметром

public static string? GetItemFromDatabase(string path,

string FullLogOrPhoneField)

{

using (StreamReader reader = new StreamReader(path))

{

string? line;

while ((line = reader.ReadLine()) != null)

{

if (line.Contains(FullLogOrPhoneField))

{

return line;

}

}

}

return null;

}

// Метод для отримання данних про товар за унікальним ідентифікатором

public static string? GetItemById(string path, string idField)

{

string[] files;

string? line;

try

{

files = Directory.GetFiles(path);

}

catch (DirectoryNotFoundException)

{

files = new string[0];

}

foreach (string filePath in files)

{

if ((line = GetItemFromDatabase(filePath, idField)) != null)

{

return line;

}

}

return null;

}

// Метод для завантаження з папки всіх продуктів з бази даних

public static LinkedList<Product> LoadData(string Path)

{

LinkedList<Product> list = new LinkedList<Product>();

string[] files;

try

{

files = Directory.GetFiles(Path);

}

catch (DirectoryNotFoundException)

{

files = new string[0];

}

foreach (string filePath in files)

{

using (StreamReader reader = new StreamReader(filePath))

{

string? Productline;

while ((Productline = reader.ReadLine()) != null)

{

string[] productProp = GetData(Productline, 9);

Product product = new Product(productProp);

list.AddLast(product);

}

}

}

return list;

}

// Метод який зчитує архів у базі даних та повертає його список

public static List<string>? LoadArchive(string Path)

{

List<string> archive = new List<string>();

try

{

using (StreamReader reader = new StreamReader(Path))

{

string? line;

while ((line = reader.ReadLine()) != null)

{

archive.Add(line);

}

}

}

catch (FileNotFoundException)

{

return null;

}

return archive;

}

// Метод для видалення або зміни змісту строки з даними у базі даних

public static void ChangeLineInDataBase(string folderPath,

string? oldLine, string? newLine, string? lineToRemove)

{

string[] files = Directory.GetFiles(folderPath);

foreach (string filePath in files)

{

string tempFilePath = Path.GetTempFileName();

using (StreamReader reader = new StreamReader(filePath))

{

using (StreamWriter writer = new StreamWriter(tempFilePath))

{

string? line;

bool lineReplaced = false;

while ((line = reader.ReadLine()) != null)

{

if (lineToRemove != null && line == lineToRemove)

{

continue;

}

else if (oldLine != null && newLine != null &&

line == oldLine && !lineReplaced)

{

writer.WriteLine(newLine);

lineReplaced = true;

}

else

{

writer.WriteLine(line);

}

}

}

}

File.Delete(filePath);

File.Move(tempFilePath, filePath);

}

}

}

}

namespace Біржа\_товарів.Models

{

// Абстрактний клас реалізує поля та конструктор

// для продавців та покупців

public abstract class User

{

public string FullName;

public string ProductsPath;

public string Archive;

public string Login;

public string Telephone;

protected string Password;

public User(string[] data)

{

FullName = data[0] + " " + data[1];

Telephone = data[2];

Login = data[3];

Password = data[4];

ProductsPath = data[5];

Archive = data[6];

}

public abstract string ClassName { get; }

}

}

namespace Біржа\_товарів.Models

{

// Клас продавця, успадковується від користувача

internal class Salesman: User

{

public override string ClassName

{

get

{

return "Продавець";

}

}

public Salesman(string[] data):base(data) { }

}

}

namespace Біржа\_товарів.Models

{

// Клас покупця, успадковується від користувача

internal class Customer: User

{

public override string ClassName

{

get

{

return "Покупець";

}

}

public Customer(string[] data): base(data) { }

}

}

namespace Біржа\_товарів.Models

{

// Клас продукта для створення екземплярів прродуктів та роботи з ними

public class Product

{

public int Id;

public string ProductName;

public int ProductAmount;

public int? ProductPrice;

public string? PaymentForm;

public string? DeliveryCondition;

public string? Adress;

public string? Notes;

public string? OwnerLogin;

public Product(string[] ProductData)

{

Id = int.Parse(ProductData[0]);

ProductName = ProductData[1];

ProductPrice = ProductData[2] != "" ? int.Parse(ProductData[2]) : 0;

ProductAmount = int.Parse(ProductData[3]);

Adress = ProductData[4];

PaymentForm = ProductData[5];

DeliveryCondition = ProductData[6];

Notes = ProductData[7];

OwnerLogin = ProductData[8];

}

// Перевизначений метод ToString() для коректного відображення знайдених продуктів

public override string ToString()

{

return $"Назва: {ProductName}; " +

$"Ціна: {ProductPrice} грн; Кількість: {ProductAmount}";

}

// Метод пошуку продуктів за заданими параметрами

public static List<Product> Search(LinkedList<Product> products,

Dictionary<string, string> searchParameters)

{

List<Product> foundProducts = new List<Product>();

foreach (Product product in products)

{

bool found = true;

foreach (var item in searchParameters)

{

string productFieldName = item.Key;

string searchParameterValue = item.Value;

var field = product.GetType().GetField(productFieldName);

var value = field?.GetValue(product);

if (field != null && field.Name == "ProductPrice" &&

value != null && value is IConvertible)

{

int actualValue = Convert.ToInt32(value);

int actualParameterValue = Convert.ToInt32(searchParameterValue);

if (actualParameterValue < actualValue)

{

found = false;

break;

}

}

else if (value != null &&

(value.ToString()?.ToLower().Contains(searchParameterValue.ToLower()) != true))

{

found = false;

break;

}

}

if (found)

{

foundProducts.Add(product);

}

}

return foundProducts;

}

}

}

namespace Біржа\_товарів

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

ApplicationConfiguration.Initialize();

Application.Run(new AuthorizationForm());

}

}

}