Міністерство освіти і науки України

Чернівецький національний університет імені Юрія Федьковича

\_\_\_\_\_\_\_Програмного забезпечення комп’ютерних систем\_\_\_\_\_\_\_

(повна назва кафедри, циклової комісії)

**КУРСОВИЙ ПРОЄКТ**

з \_\_Принципи конструювання програмного забезпечення\_\_

(назва дисципліни)

на тему:\_\_\_ **Система управління магазином з одним продавцем** \_\_\_\_\_\_\_\_\_

Виконав: студент (ка) \_\_2\_\_\_ курсу, групи\_243Б\_\_

спеціальності \_\_\_\_\_121\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр спеціальності)

\_\_\_\_Інженерія програмного забезпечення\_\_\_\_\_

(назва спеціальності)

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_Шилов Б.В.\_\_\_\_

(підпис) (прізвище, ініціали)

Керівник \_\_\_\_\_\_\_Комісарчук В.В \_\_\_

(прізвище, ініціали)

|  |  |  |  |
| --- | --- | --- | --- |
| **До захисту допущено:** |  |  | **Дата захисту** «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_р. |
| від «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_р. |  |  | **Оцінка:** |
| Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |  | за національною шкалою \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (підпис) |  |  | (словами) |
| Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |  | кількість балів \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (підпис) |  |  | (цифра) |
|  |  |  | за шкалою ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  | (літера) |

Чернівці, 2022

Чернівецький національний університет імені Юрія Федьковича

Кафедра \_Програмного забезпечення комп’ютерних систем\_\_\_

Спеціальність \_Інженерія програмного забезпечення\_\_\_\_\_\_\_\_\_\_\_\_\_

**ЗАВДАННЯ**

**на курсовий проєкт студенту**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шилова Богдана Вікторовича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по-батькові)

1. Тема проєкту:\_\_\_\_\_Система управління магазином з одним продавцем \_\_\_

2. Вихідні дані до проєкту:

– розробити додаток;

–розробити функціонал(реєстрація надходження товару, оформлення покупки, проблема уцінки і списання, інвентаризація залишків товару з обчисленням сумарної вартості);

3. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити):

– розробити загальні вимоги до програми;

– описати модулі програми та алгоритм роботи у вигляді блок-схеми;

– описати методи програми;

– опис користувацького інтерфейсу;

– зробити висновок;

– навести текст програми у вигляді програмного коду

4. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень):

\_– блок-схеми роботи з програмою;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_– скріншоти роботи з програмою.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Завдання прийняв (ла) до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис студента)

Керівник проєкту \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис керівника)

**РЕФЕРАТ**

В курсовому проєкті розроблено додаток “Магазин”.

Програма розрахована на користувачів, які не мають спеціальної комп’ютерної підготовки. Досвід роботи не вимагається.

# Область застосування – програмне забезпечення (додаток) для не дорогих ПК.

Розробка реалізована засобами середовища Visual Studio на мові C# із застосуванням графічної програми Windows Forms. Дане середовище є зручним у використанні для швидкого та якісного створення додатків на OC Windows.

Дана розробка у майбутньому може бути розширена із добавленням нового функціоналу і видозміненою логікою обробки.

Курсовий проект містить: 54 с., 13 рис., 9 табл., 1 додаток, 3 джерел.

**SUMMARY**

In the course project, the application "Shop" was developed.

The program is intended for users who do not have special computer training. Work experience is not required.

The field of application is software (applications) for inexpensive PCs.

The development was implemented using the Visual Studio environment in the C# language using the Windows Forms graphic program. This environment is convenient to use for quick and high-quality creation of bases on OC Windows.

This development can be expanded in the future with the addition of new functionality and modified processing logic.

The course project contains: 54 pages, 13 figures, 9 tables, 1 appendix, 3 sources..

**ЗМІСТ**

1. АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ…………..….……6
   1. Загальні вимоги до програми…………………………………………….6

1.1.1 Вимоги до графічного інтерфейсу користувача……………………6

1.1.2 Вимоги до архітектури програми……………………………………6

1.1.3 Вимоги до функціональності додатка………………………………7

1.2.Призначення та область застосування………………………….…………...7

1.3Функціональні вимоги……………………………………………………...8-9

1. ОПИС ПРОГРАМИ…………………………………………………….……..9
   1. Структура програми………………………………………………………9

2.1.1 Модулі програми……………………………………………………..9

2.1.2 Алгоритми роботи програми……………………………………….10

* 1. Опис методів програми………………………………………………….11
  2. Програмні засоби………………………………………………...………15
  3. Опис користувацького інтерфейсу……………………………………...16

ВИСНОВКИ……………………………………………………………...………18

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……………………………....….……17

ДОДАТКИ………………………………………………………………..………17

Додаток А. Скролінг (текст) програми………………………………….18-29

1. **АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ**

**1.1 Загальні вимоги до програми**

* + 1. **Вимоги до графічного інтерфейсу користувача**

− мова програмування (C#, Python) – на вибір студента.

− програмний продукт – віконний (graphical user interface) додаток.

− вхідні дані подаються користувачем на GUI.

− результати відображаються користувачу у віконному режимі.

− база даних – це текстовий файл, тому передбачити збереження даних у файл та зчитування даних з файлу.

− передбачити навігаційну панель (меню), яка повинна забезпечувати перегляд, редагування, додавання, видалення даних.

− передбачити сортування, фільтрацію та пошук даних по самостійно вибраному ключу.

− обов’язкова перевірка коректності вводу даних та обробка виключень.

− наявність «посібника користувача» – інструкції з використання програми.

* + 1. **Вимоги до архітектури програми**

Використання стандартних контейнерних класів які реалізують:

− вибір дати;

– вибір продукту;

– вибір масиву даних;

* + 1. **Вимоги до функціональності додатка**

1) Відкриття головного вікна з кнопками вибору дії.

2) Вибір кнопки(дії).

3) У залежності від вибраної кнопки виконуються дії:

**•** реєстрація товару

• фільтрування за умовами

• додавання до кошику

• продаж

• пошук

• списання

• очищення масиву даних

• збереження даних в файл

• зчитування даних з файлу

• оновлення(перезапис, додавання) даних в файлі

• обрахунок загальної вартості

• скидання фільтрів

4) Повернення до головного вікна

1.2 Призначення та область застосування

Мета роботи полягає у розробці бази даних “Магазин ”для користувачів ОС Windows.

Реалізована версія “Магазин” призначена для ділового використання користувачами, що працюють під управлінням ОС Windows, при роботі.

Область застосування – програмне забезпечення для комп’ютерів з

обмежними обчислювальними ресурсами.

**1.3 Функціональні вимоги**

До програмного забезпечення висуваються такі функціональні вимоги:

1. Програмне забезпечення повинно забезпечувати роботу користувача одночасно з усіма потрібними функціями шляхом створення додаткових форм або розділенням робочої частини програми.

2. Кнопки, які повинна забезпечувати програма:

|  |  |
| --- | --- |
|  | Додавання |
|  | Пошук |
|  | Інструменти |
|  | Продаж |
|  | Файл |
|  | Збереження |
|  | Оновлення |
|  | Завантаження |
|  | Корзина |
|  | Про нас |
|  | Допомога |

2.1. Якщо обрали і натиснули кнопку, має відкритися відповідне вікно.

2.2. При повторному натисненні усіх кнопок окрім збереження, оновлення та завантаження файлу, відповідне вікно закривається.

2.3. Колір піктограми вказує, чи відкрите вікно(область). Зелений – відкрите, червоний - закрите.

3. Функціональні вимоги до додавання:

3.1. Повинна з’явитись область з можливість реєстрації нового продукту.

4. Функціональні вимоги до пошуку:

4.1. Повинна з’явитись область з можливістю пошуку або списання товару.

5. Функціональні вимоги до інструментів:

5.1. Повинна з’явитись область, де можна встановити фільтри, скинути їх, очистити масив даних, порахувати загальну ціну.

6. Функціональні вимоги до продажу:

6.1. Повинна з’явитись область, де можна додати товар до корзини та продати всі товари, що знаходяться в корзині.

7. Функціональні вимоги до файлу:

7.1. Повинне відкритись підменю, де можна обрати збереження, оновлення чи завантаження файлу.

8. Функціональні вимоги до збереження:

8.1. Повинен відкритись діалог для вибору файлу для збереження даних, та відповідно збереження даних в цей файл.

9. Функціональні вимоги до оновлення:

9.1. Повинен відкритись діалог для вибору файлу для оновлення даних, та відповідно оновлення даних(додавання даних з масиву в програмі до тих, що вже були в файлі).

10. Функціональні вимоги до завантаження:

10.1. Повинен відкритись діалог для вибору файлу для завантаження, дані якого буде завантажено в програму та відображено.

11. Функціональні вимоги до корзини:

11.1. Повинна відкритись нова форма(вікно) в якій буде відображено продукти, що знаходяться в корзині.

12. Функціональні вимоги до кнопки про нас:

12.1. Повинна відкритись нова форма(вікно) в якій буде коротка інформація про виробника.

13. Функціональні вимоги до допомоги:

13.1. Повинна відкритись нова форма(вікно) в якій буде коротка інформація про роботу з програмою.

**2. ОПИС ПРОГРАМИ**

**2.1 Структура програми**

**2.1.1 Модулі програми**

Робота розробленого програмного забезпечення реалізується наступними модулями:

1) Date – клас для збереження та обробки дати.

2) Product – клас продукту, в якому збережено дані про кожен окремий продукт.

3) Products – клас який є масивом даних в програмі. Реалізує роботу з файлом.

4) EmptyDataException – абстрактний клас для реалізації загальної логіки виключень.

5) EmptySearchDataException – клас-виключення для недостатньої кількості данихпри пошуку.

6) IncorrectRegisterDataException – клас-виключення для некоректних даних при реєстрації продукту.

7) IncorrectWriteOffDataException - клас-виключення для некоректних даних при списуванні.

8) IncorrectSearchDataException – клас-виключення для некоректних даних при пошуку.

9) IncorrectFilterDataException – клас-виключення для некоректних даних для фільтрації.

10) IncorrectSellDataException – клас-виключення для некоректних даних для продажу(додавання до корзини).

11) EmptyProductsException – клас-виключення для відсутності даних при спробі до них звернутись.

12) IncorrectDataException – клас-виключення для некоректних даних в файлі при зчитуванні.

13) NotEnoughAmount – клас-виключення для недостатньої кількості товару при спробі списати/продати.

14)Form1 – клас-форма. Головне вікно програми.

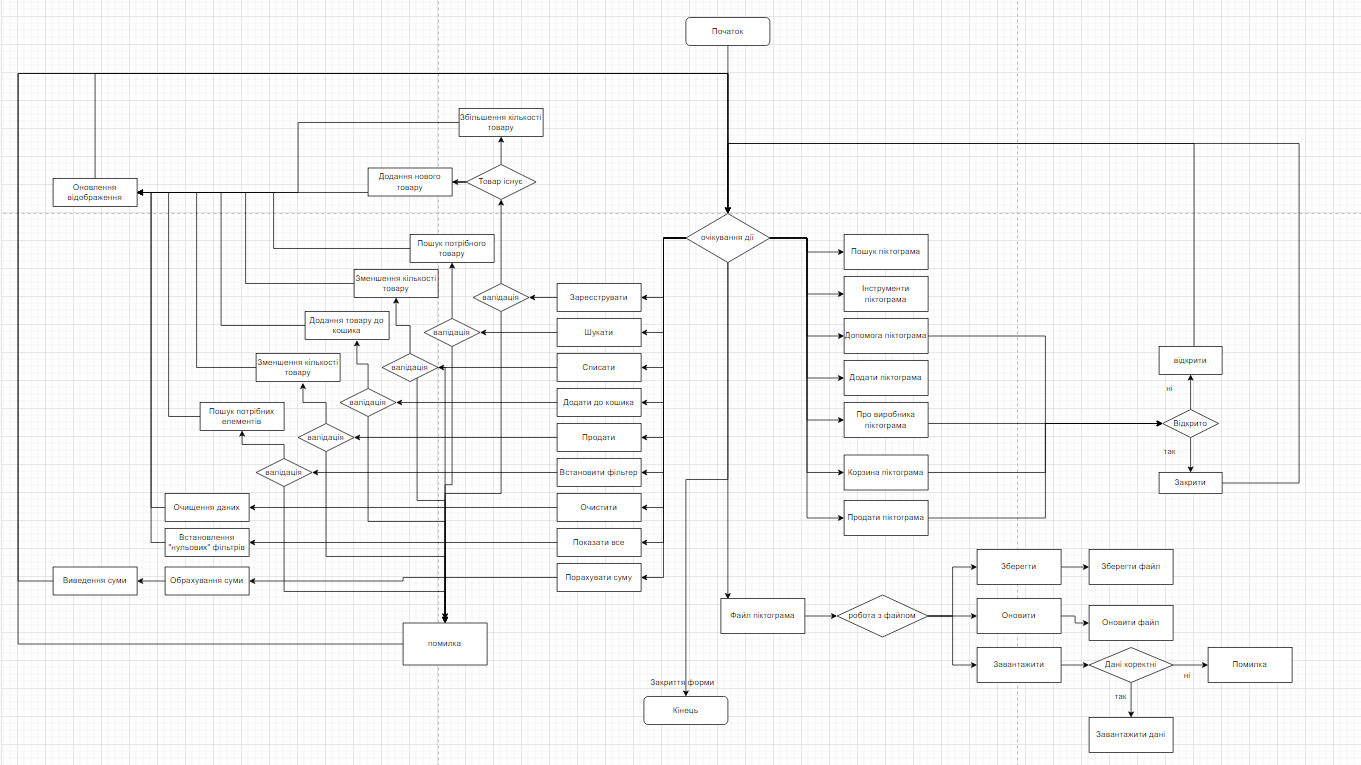
15)Basket – клас-форма. Вікно для відображення вмісту кошика.

16)Bill – клас-форма. Вікно для відображення рахунку при продажі товару. Чек може бути пустим.

17)AboutUs – клас-форма. Вікно для відображення інформації про виробника.

18)Help – клас-форма. Вікно для відображення інформації про користування програмою.

**2.1.2 Алгоритми роботи програми**

****

**2.2 Опис методів програми**

Список методів класу Date та їх опис наведено в табл.1.

Список методів класу Product та їх опис наведено в табл.2.

Список методів класу Products та їх опис наведено в табл.3.

Список методів класів-виключень та їх опис наведено в табл.4.

Список методів класу Form1 та їх опис наведено в табл.5.

Список методів класу Basket та їх опис наведено в табл.6.

Список методів класу Bill та їх опис наведено в табл.7.

Список методів класу AboutUs та їх опис наведено в табл.8.

Список методів класу Help та їх опис наведено в табл.9.

Таблиця 1 – Основні методи класу Date

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | *public static bool operator <(Date, Date),*  *public static bool operator >(Date, Date)* | Оператор порівняння дат |

Таблиця 2 – Основні методи класу Product

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | *Клас не має методів* |  |

Таблиця 3 – Основні методи класу Products

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | *public void Add\_product(Product)* | Додає продукт до списку |
| 2. | *public void*  *Delete\_product(Product)* | Видаляє продукт з списку |
| 3. | *public void Read\_file(string)* | Зчитує файл та вносить до списку вмістиме |
| 4. | *public void Save\_file(string)* | Записує список продуктів в файл |
| 5. | *private void write\_data(StreamWriter, object)* | Метод для зручнішого запису в файл. |
| 6. | *public void Add\_file(string)* | Оновлює файл, зчитуючи з нього інформацію, додаючи інформацію в програмі та записує нові дані в файл |

Таблиця 4 – Основні методи класів-виключень

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | *public override void ShowMessageBox()* | Виводить MassegeBox(вікно інформації) з відповідним текстом |

Таблиця 5 – Основні методи класу Form1

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | *private void Search\_button\_Click(objec, EventArgs)* | Викликається при натисненні кнопки шукати. Залишає в основній частині програми відображеними тільки дані, що відповідають пошуку |
| 2. | *private void addMenuItem\_Click(object, EventArgs)* | Викликається при натисненні піктограми додати. Відкриває або закриває область з реєстрацією товару |
| 3. | *private void searchMenuItem\_Click(object, EventArgs)* | Викликається при натисненні піктограми шукати. Відкриває або закриває область з пошуком/списанням продуктів |
| 4. | *private void filterMenuItem\_Click(object, EventArgs)* | Викликається при натисненні піктограми інструменти. Відкриває або закриває область з інструментами для встановлення або скидання фільтрів, підрахунку суми товарів та очищення дпних |
| 5. | *private void sellMenuItem\_Click(object, EventArgs)* | Викликається при натисненні піктограми продати. Відкриває або закриває область для додавання до корзини або продажу товарів з корзини |
| 6. | *private void saveToolStripMenuItem\_Click(object, EventArgs)* | Викликається при натисненні піктограми зберегти. Відкриває діалогове вікно для зберігання та зберігає дані в файл |
| 7. | *private void saveAddToolStripMenuItem\_Click(object, EventArgs)* | Викликається при натисненні на піктограму оновити. Відкриває діалогове вікно для вибору файлу та оновлює файл |
| 8. | *private void loadToolStripMenuItem\_Click(object, EventArgs)* | Викликається при натисненні піктограми завантажити. Відкриває діалогове вікно для обрання файлу та завантажує дані з файлу до програми |
| 9. | *private void UpdateDataGrid()* | Оновлює відображення даних |
| 10. | *private void Add\_product\_To\_DataGridView(Product)* | Додає продукт до відображення |
| 11. | *private void Register\_button\_Click(object, EventArgs)* | Викликається при натисненні кнопки реєстрації. Додає товар до списку. Якщо такий товар існує – збільшує кількість цього товару |
| 12. | *private void Show\_all\_button\_Click(object, EventArgs)* | Викликається при натисненні кнопки показати все. Зупиняє дію обмежень виведення інформації по пошуку або по фільтру |
| 13. | *private void filter\_button\_Click(object, EventArgs)* | Викликається при натисканні кнопки застосувати фільтр. Встановлює значення фільтру та оновлює виведення даних у відповідності до нього |
| 14. | *private void Write\_off\_button\_Click(object, EventArgs)* | Викликається при натисненні кнопки списати. Якщо такий товар існує, зменшує його кількість на вказану. Якщо при цьому кількість стане рівною нулю, товар зникне |
| 15. | *private void Count\_sum\_button\_Click(object, EventArgs)* | Викликається при натисненні кнопки порахувати суму. Обчислює суму всіх продуктів, що зазначенні в програмі та виводить MessageBox(вікно інформації) для відображення |
| 16. | *private void Add\_basket\_button\_Click(object, EventArgs)* | Викликається при натисненні кнопки додати до корзини. Додає вказаний продукт до корзини, та якщо відкрите вікно корзини, оновлює виведенні дані |
| 17. | *private void Sell\_button\_Click(object, EventArgs)* | Викликається при натисненні кнопки продати. Якщо продуктів достатньо, зменшує їхню кількість на ту, що знаходиться в корзині. Якщо кількість продукту в результаті стає нуль, продукт зникає. Опісля очищує корзину, оновлює виведення даних в головному вікні та у вікні корзини |
| 18. | *private void basketMenuItem\_Click(object, EventArgs)* | Викликається при натисненні піктограми корзмна. Відкриває або закриває форму(вікно) із вмістом корзини. |
| 19. | *public void basket\_closing()* | Викликається при закритті форми(вікна) корзини. Змінює колір піктограми корзини на червоний |
| 20. | *public void about\_us\_closing()* | Викликається при закритті форми(вікна) інформації про виробника. Заміняє піктограму про виробника на червону |
| 21. | *private void aboutUsMenuItem\_Click(object, EventArgs)* | Викликається при натисненні на піктограму про виробника. Відкриває або закриває форму(вікно) з інформацією про виробника |
| 22. | *public void help\_closing()* | Викликається при закритті форми(вікна) з допомогою. Заміняє піктограму допомоги на червону |
| 23. | *private void helpMenuItem\_Click(object, EventArgs)* | Викликається при натисненні на піктограму допомоги. Відкриває або закриває форму(вікно) з інформацією про програму |
| 24. | *private void clear\_button\_Click(object, EventArgs)* | Викликається при натисненні на кнопку очистити. Видаляє дані про продукти та корзину з програми та оновлює відображення даних. |

Таблиця 6 – Основні методи класу Basket

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | *public void updateBasketDataGrid(Products)* | Оновлює відображення продуктів в корзині у формі(вікні) |
| 2. | *private void Add\_product\_To\_DataGridView(Product)* | Додає продукт до відображення в корзині |
| 3. | *private void Basket\_FormClosing(object, FormClosingEventArgs)* | Викликається при закритті форма(вікна) корзини. Викликає функцію заміни піктограми в головному вікні |

Таблиця 7 – Основні методи класу Bill

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | *private void ok\_button\_Click(object, EventArgs)* | Викликається при натисненні кнопки ок. Закриває фірму(вікно) чеку |

Таблиця 8 – Основні методи класу AboutUs

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | *private void ok\_button\_Click(object, EventArgs)* | Викликається при натисненні кнопки ок. Закриває форму(вікно) про виробників |
| 2. | *private void AboutUs\_FormClosing(object, FormClosingEventArgs)* | Викликається при закритті форми(вікна) про виробників. Викликає функцію заміни піктограми в головному вікні |

Таблиця 9 – Основні методи класу Help

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | *private void ok\_button\_Click(object, EventArgs)* | Викликається при натисненні кнопки ок. Закриває форму(вікно) з допомогою |
| 2. | *private void Help\_FormClosing(object, FormClosingEventArgs)* | Викликається при закритті форми(вікна) з допомогою. Викликає функцію заміни піктограми в головному вікні |

**2.3 Програмні засоби**

Розробка реалізована засобами середовища Visual Studio на мові C#. Дане середовище є зручним у використанні для швидкого та якісного створення додатків на ОС Windows.

Windows забезпечує повний набір програмного забезпечення для комп’ютерів за його основою: операційну систему, сполучні програми і набори для розробників.

Windows – це операційна система, яка дозволяє розробникам створювати програми, які можуть повною мірою скористатись функціями комп’ютера за її керуванням. Наприклад, програма може звернутися до будь-якої складової комп’ютера (дисплею, камери, мікрофона, мишки та клавіатури), що дозволяє розробникам створювати більш багаті і інтуїтивно зрозумілі для користувачів програми.

**2.4 Опис користувацького інтерфейсу**

При запуску програми, користувач має можливість обрати кнопку для виконання певної дії. Після натискання з’являється нове вікно(область) на екрані, де розташована різна інформація, кнопки (в залежності від обраної функції).

Розглянемо дії користувача при роботі з програмою. Користувач запускає додаток . На екрані комп’ютера з’являється головне вікно програми (рис.12).

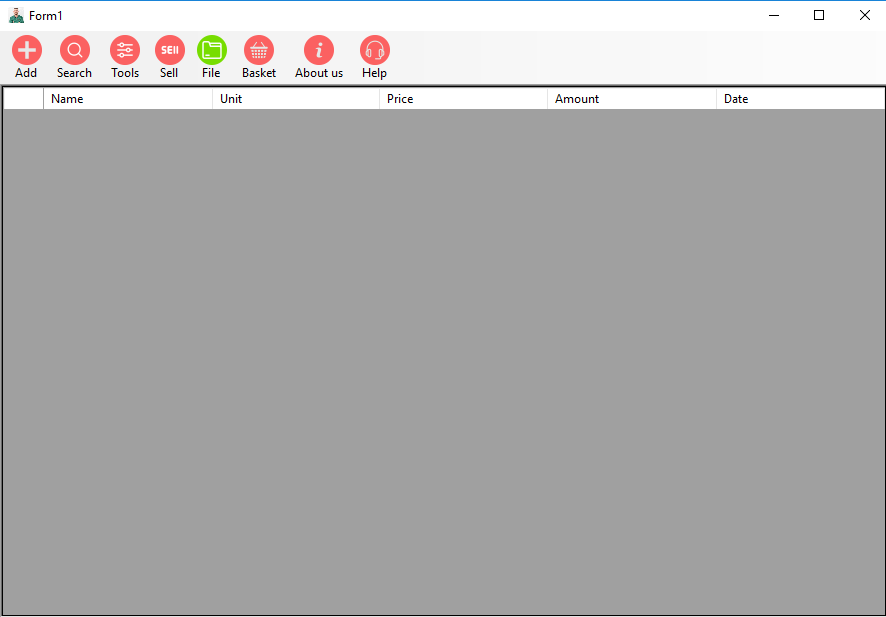
****

Рисунок 12 – Головне вікно програми

Тут ми бачимо чотири основні кнопки для вибору дії в шапці програми. Нажимаючи на потрібні піктограми, можна обрати потрібні області. При Включенні всіх вікон та областей програма виглядає наступним чином

(рис 13)

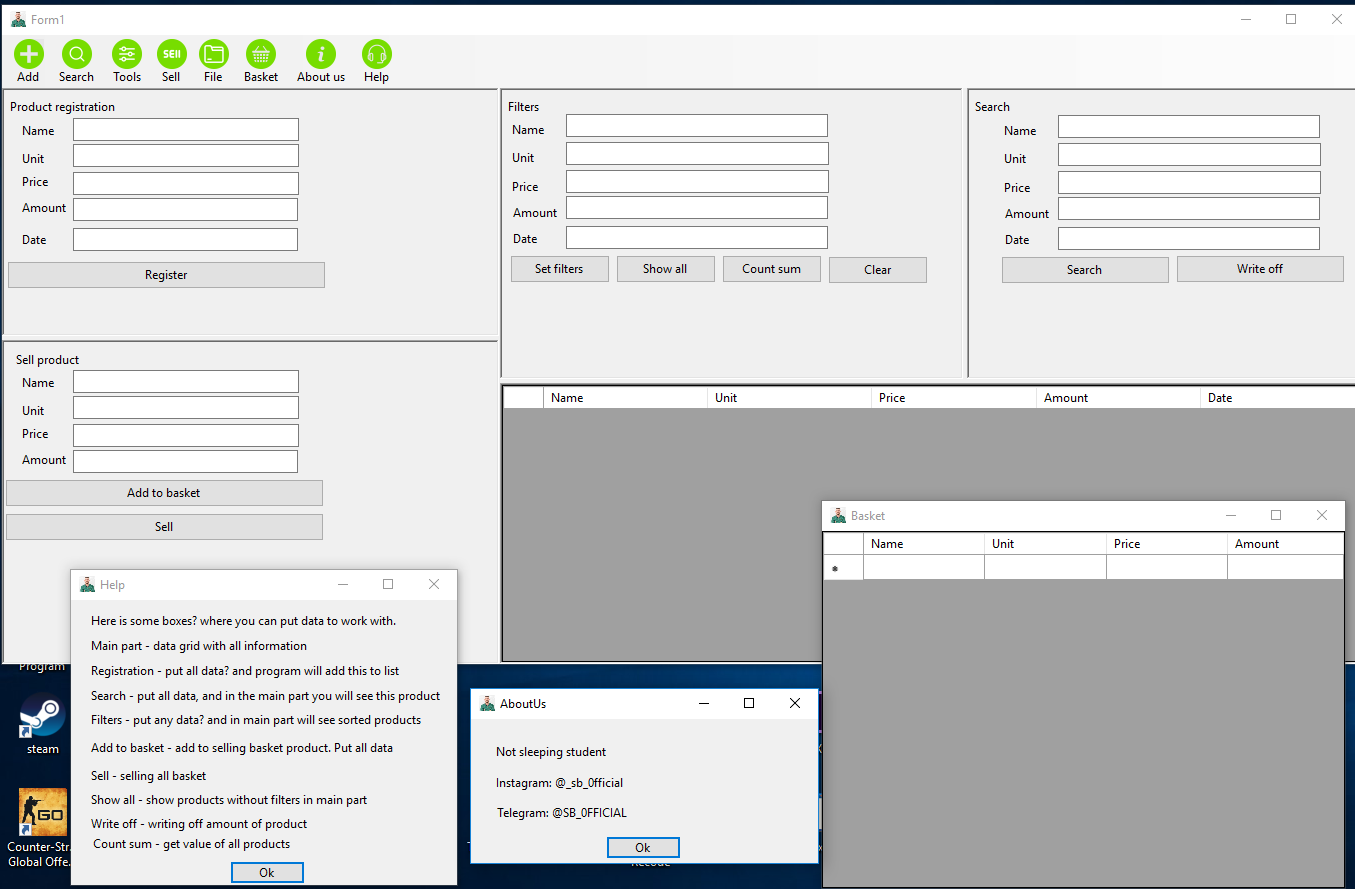


Рисунок 13 – Повне вікно програми

Поля схожі між собою, і практично повторюються. Але це зроблено задля можливості паралельного виконання потрібних дій без постійного переписування даних.

**ВИСНОВКИ**

В результаті виконання курсового проєкту розроблена система управління базою даних “Магазин”під ОС Windows.

Розроблене програмне забезпечення має наступні можливості:

• Зберігати, оновлювати та завантажувати дані з файлу.

• Здійснити реєстрацію та продаж товарів.

В даній роботі було застосовано Windows Forms – графічну (презентаційну)

підсистему, яка використовується для створення сучасних інтерфейсів користувача за керуванням ОС Windows.

Дана розробка у майбутньому може бути розширена із додаванням нового функціоналу.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1.<https://stackoverflow.com/>

2. <https://metanit.com/>

3.<https://www.flaticon.com/>

**ДОДАТКИ**

**ДОДАТОК А. Скролінг (текст) програми**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Thesis

{

public partial class Help : Form

{

private Form1 parent;

public Help(Form1 p)

{

this.parent = p;

InitializeComponent();

}

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

{

this.Close();

}

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

{

this.parent.help\_closing();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Thesis

{

public partial class Bill : Form

{

public Bill(string prod\_text,double sum)

{

InitializeComponent();

this.Products\_text.Text = "Name\tUnit\tPrice\tAmount\n";

this.Products\_text.Text += prod\_text;

this.Sum.Text = sum.ToString();

}

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

{

this.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Thesis

{

public partial class Basket : Form

{

private Form1 parrent;

public Basket(Form1 p)

{

parrent = p;

InitializeComponent();

this.dataGridView1.ReadOnly = true;

}

public void updateBasketDataGrid(Products prod)

{

this.dataGridView1.Rows.Clear();

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

{

this.Add\_product\_To\_DataGridView(prod.Data[i]);

}

}

private void Add\_product\_To\_DataGridView(Product prod)

{

this.dataGridView1.Rows.Add(prod.Name, prod.Unit, prod.Price.ToString(), prod.Amount.ToString());

}

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

{

parrent.basket\_closing();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

namespace Thesis

{

public partial class AboutUs : Form

{

private Form1 parent;

public AboutUs(Form1 p)

{

this.parent = p;

InitializeComponent();

}

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

{

this.Close();

}

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

{

this.parent.about\_us\_closing();

}

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace Thesis

{

public class Products

{

private List<Product> data = new List<Product>();

public void Add\_product(Product add\_product)

{

this.data.Add(add\_product);

}

public void Delete\_product(Product del\_product)

{

this.data.Remove(del\_product);

}

public void Read\_file(string path)

{

try

{

this.data = new List<Product>();

StreamReader reader = new StreamReader(path);

string line = reader.ReadLine();

int i = 0;

while (line != "" && line != null)

{

this.data.Add(new Product());

string[] words = line.Split(' ');

if (words.Length != 7)

{

reader.Close();

throw new IncorrectDataException();

}

this.data[i].Name = words[0];

this.data[i].Unit = words[1];

double read\_price = 0;

if (double.TryParse(words[2], out read\_price))

{

this.data[i].Price = read\_price;

}

else

{

reader.Close();

throw new IncorrectDataException();

}

int read\_amount = 0;

if (int.TryParse(words[3], out read\_amount))

{

this.data[i].Amount = read\_amount;

}

else

{

reader.Close();

throw new IncorrectDataException();

}

int read\_day;

if (int.TryParse(words[4], out read\_day))

{

this.data[i].Date.Day = read\_day;

}

else

{

reader.Close();

throw new IncorrectDataException();

}

int read\_month = 0;

if (int.TryParse(words[5], out read\_month))

{

this.data[i].Date.Month = read\_month;

}

else

{

reader.Close();

throw new IncorrectDataException();

}

int read\_year = 0;

if (int.TryParse(words[6], out read\_year))

{

this.data[i].Date.Year = read\_year;

}

else

{

reader.Close();

throw new IncorrectDataException();

}

line = reader.ReadLine();

i++;

}

reader.Close();

}

catch (IncorrectDataException exept)

{

exept.ShowMessageBox();

}

}

public void Save\_file(string path)

{

StreamWriter writer = new StreamWriter(path);

this.data.ForEach(product => {

this.write\_data(writer, product.Name);

this.write\_data(writer, product.Unit);

this.write\_data(writer, product.Price);

this.write\_data(writer, product.Amount);

this.write\_data(writer, product.Date.Day);

this.write\_data(writer, product.Date.Month);

writer.Write(product.Date.Year);

writer.WriteLine();

});

writer.Close();

}

private void write\_data(StreamWriter writer, object write\_data)

{

writer.Write(write\_data);

writer.Write(" ");

}

public void Add\_file(string path)

{

try

{

List<Product> data\_read = new List<Product>();

StreamReader reader = new StreamReader(path);

string line = reader.ReadLine();

int i = 0;

while (line != "" && line != null)

{

data\_read.Add(new Product());

string[] words = line.Split(' ');

if (words.Length != 7)

{

reader.Close();

throw new IncorrectDataException();

}

data\_read[i].Name = words[0];

data\_read[i].Unit = words[1];

double read\_price = 0;

if (double.TryParse(words[2], out read\_price))

{

data\_read[i].Price = read\_price;

}

else

{

reader.Close();

throw new IncorrectDataException();

}

int read\_amount = 0;

if (int.TryParse(words[3], out read\_amount))

{

data\_read[i].Amount = read\_amount;

}

else

{

reader.Close();

throw new IncorrectDataException();

}

int read\_day;

if (int.TryParse(words[4], out read\_day))

{

data\_read[i].Date.Day = read\_day;

}

else

{

reader.Close();

throw new IncorrectDataException();

}

int read\_month = 0;

if (int.TryParse(words[5], out read\_month))

{

data\_read[i].Date.Month = read\_month;

}

else

{

reader.Close();

throw new IncorrectDataException();

}

int read\_year = 0;

if (int.TryParse(words[6], out read\_year))

{

data\_read[i].Date.Year = read\_year;

}

else

{

reader.Close();

throw new IncorrectDataException();

}

line = reader.ReadLine();

i++;

}

reader.Close();

for(int j = 0; j < this.data.Count; j++)

{

bool searched = false;

for(int k = 0; k < data\_read.Count; k++)

{

if(this.data[j].Name == data\_read[k].Name && this.data[j].Unit == data\_read[k].Unit && this.data[j].Price == data\_read[k].Price && this.data[j].Date.Day == data\_read[k].Date.Day && this.data[j].Date.Month == data\_read[k].Date.Month && this.data[j].Date.Year == data\_read[k].Date.Year)

{

data\_read[k].Amount += this.data[j].Amount;

searched = true;

break;

}

}

if (!searched)

{

data\_read.Add(this.data[j]);

}

}

StreamWriter writer = new StreamWriter(path);

data\_read.ForEach(product => {

this.write\_data(writer, product.Name);

this.write\_data(writer, product.Unit);

this.write\_data(writer, product.Price);

this.write\_data(writer, product.Amount);

this.write\_data(writer, product.Date.Day);

this.write\_data(writer, product.Date.Month);

writer.Write(product.Date.Year);

writer.WriteLine();

});

writer.Close();

}

catch (IncorrectDataException exept)

{

exept.ShowMessageBox();

}

}

public int Length

{

get

{

if (this.data == null)

{

return 0;

}

return this.data.Count;

}

}

public List<Product> Data { get { return this.data; } }

}

}

using System;

using System.Collections.Generic;

using System.Text;

namespace Thesis

{

public class Product

{

private string name;

private string unit;

private double price;

private int amount;

private Date date = new Date();

public string Name { get { return this.name; } set { this.name = value; } }

public string Unit { get { return this.unit; } set { this.unit = value; } }

public double Price { get { return this.price; } set { this.price = value; } }

public int Amount { get { return this.amount; } set { this.amount = value; } }

public Date Date { get { return this.date; } set { this.date = value; } }

}

}

using System;

using System.Collections.Generic;

using System.Text;

using System.Windows.Forms;

namespace Thesis

{

public abstract class EmptyDataException : Exception

{

public abstract void ShowMessageBox();

}

public class EmptySearchDataException : EmptyDataException

{

public EmptySearchDataException() : base() { }

public override void ShowMessageBox()

{

MessageBox.Show("Data is empty", "Incorrect data", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public class IncorrectRegisterDataException : EmptyDataException

{

public IncorrectRegisterDataException() : base() { }

public override void ShowMessageBox()

{

MessageBox.Show("Incorrect register data", "Incorrect data", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public class IncorrectWriteOffDataException : EmptyDataException

{

public IncorrectWriteOffDataException() : base() { }

public override void ShowMessageBox()

{

MessageBox.Show("Incorrect write off data", "Incorrect data", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public class IncorrectSearchDataException : EmptyDataException

{

public IncorrectSearchDataException() : base() { }

public override void ShowMessageBox()

{

MessageBox.Show("Incorrect search data", "Incorrect data", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public class IncorrectFilterDataException : EmptyDataException

{

public IncorrectFilterDataException() : base() { }

public override void ShowMessageBox()

{

MessageBox.Show("Incorrect filter data", "Incorrect data", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public class IncorrectSellDataException : EmptyDataException

{

public IncorrectSellDataException() : base() { }

public override void ShowMessageBox()

{

MessageBox.Show("Incorrect sell data", "Incorrect data", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public class EmptyProductsException : EmptyDataException

{

public EmptyProductsException() : base() { }

public override void ShowMessageBox()

{

MessageBox.Show("No one product is here", "Incorrect data", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public class IncorrectDataException : Exception

{

public void ShowMessageBox()

{

MessageBox.Show("Incorrect data in file", "Incorrect data", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

public class NotEnoughAmount : Exception

{

public void ShowMessageBox()

{

MessageBox.Show("Not enough amount", "Incorrect data", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

using System;

using System.Collections.Generic;

using System.Text;

namespace Thesis

{

public class Date

{

private int day;

private int month;

private int year;

public static bool operator <(Date a, Date b)

{

if (a.year < b.year)

{

return true;

}

else if (a.year == b.year)

{

if (a.month < b.month)

{

return true;

}

else if (a.month == b.month)

{

if (a.Day < b.Day)

{

return true;

}

else

{

return false;

}

}

else

{

return false;

}

}

else

{

return false;

}

}

public static bool operator >(Date a, Date b)

{

if (a.year > b.year)

{

return true;

}

else if (a.year == b.year)

{

if (a.month > b.month)

{

return true;

}

else if (a.month == b.month)

{

if (a.Day > b.Day)

{

return true;

}

else

{

return false;

}

}

else

{

return false;

}

}

else

{

return false;

}

}

public int Day { get { return this.day; } set { this.day = value; } }

public int Month { get { return this.month; } set { if (value <= 12) { this.month = value; } } }

public int Year { get { return this.year; } set { this.year = value; } }

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace Thesis

{

public partial class Form1 : Form

{

private Basket Basket\_window;

private TextBox[] SearchBoxes;

private TextBox[] RegisterBoxes;

private TextBox[] FilterBoxes;

private TextBox[] SellBoxes;

private Product SearchData = new Product();

private Product FilterData = new Product();

private Products products = new Products();

private Products basket = new Products();

private Product SellData = new Product();

private bool addBoxOpen = false;

private bool filterBoxOpen = false;

private bool searchBoxOpen = false;

private bool sellBoxOpen = false;

private bool helpOpen = false;

private bool basketOpen = false;

private bool filterOn = false;

private bool searchOn = false;

private bool aboutOpen = false;

private Bill bill;

private AboutUs aboutUs;

private Help help;

public Form1()

{

InitializeComponent();

this.dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

this.dataGridView1.AllowUserToAddRows = false;

this.SearchBoxes = new TextBox[5];

this.RegisterBoxes = new TextBox[5];

this.FilterBoxes = new TextBox[5];

this.SellBoxes = new TextBox[4];

this.SearchBoxes[0] = this.Search\_name\_data;

this.SearchBoxes[1] = this.Search\_unit\_data;

this.SearchBoxes[2] = this.Search\_price\_data;

this.SearchBoxes[3] = this.Search\_amount\_data;

this.SearchBoxes[4] = this.Search\_date\_data;

this.RegisterBoxes[0] = this.Registration\_name\_data;

this.RegisterBoxes[1] = this.Registration\_unit\_data;

this.RegisterBoxes[2] = this.Registration\_price\_data;

this.RegisterBoxes[3] = this.Registration\_amount\_data;

this.RegisterBoxes[4] = this.Registration\_date\_data;

this.FilterBoxes[0] = this.Filter\_name\_data;

this.FilterBoxes[1] = this.Filter\_unit\_data;

this.FilterBoxes[2] = this.Filter\_price\_data;

this.FilterBoxes[3] = this.Filter\_amount\_data;

this.FilterBoxes[4] = this.Filter\_date\_data;

this.SellBoxes[0] = this.Sell\_name\_data;

this.SellBoxes[1] = this.Sell\_unit\_data;

this.SellBoxes[2] = this.Sell\_price\_data;

this.SellBoxes[3] = this.Sell\_amount\_data;

this.dataGridView1.ReadOnly = true;

this.addMenuItem.Image = Properties.Resources.add\_closed;

this.filterMenuItem.Image = Properties.Resources.adjust\_closed;

this.searchMenuItem.Image = Properties.Resources.search\_closed;

this.sellMenuItem.Image = Properties.Resources.sell\_closed;

this.basketMenuItem.Image = Properties.Resources.basket\_closed;

this.aboutUsMenuItem.Image = Properties.Resources.about\_us\_closed;

this.helpMenuItem.Image = Properties.Resources.help\_closed;

this.splitContainer1.Panel1.Enabled = false;

this.splitContainer2.Panel1.Enabled = false;

this.splitContainer3.Panel1.Enabled = false;

this.splitContainer3.Panel2.Enabled = false;

this.splitContainer4.Panel1.Enabled = false;

this.splitContainer4.Panel2.Enabled = false;

this.splitContainer2.Panel1Collapsed = true;

this.splitContainer3.Panel1Collapsed = true;

this.splitContainer3.Panel2Collapsed = true;

this.splitContainer1.Panel1Collapsed = true;

this.splitContainer4.Panel1Collapsed = true;

this.splitContainer4.Panel2Collapsed = true;

this.splitContainer2.SplitterDistance = this.splitContainer2.Width / 5;

this.splitContainer2.Panel1.AutoScroll = true;

}

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

{

try

{

bool dataEmpty = false;

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

{

if(SearchBoxes[i].Text == "")

{

dataEmpty = true;

break;

}

}

if (dataEmpty)

{

throw new EmptySearchDataException();

}

try

{

this.SearchData.Name = this.SearchBoxes[0].Text;

this.SearchData.Unit = this.SearchBoxes[1].Text;

double search\_price = 0;

if (double.TryParse(this.SearchBoxes[2].Text, out search\_price))

{

this.SearchData.Price = search\_price;

}

else

{

throw new IncorrectSearchDataException();

}

int search\_amount = 0;

if (int.TryParse(this.SearchBoxes[3].Text, out search\_amount))

{

this.SearchData.Amount = search\_amount;

}

else

{

throw new IncorrectSearchDataException();

}

string[] day\_words = this.SearchBoxes[4].Text.Split('.');

if (day\_words.Length != 3)

{

throw new IncorrectSearchDataException();

}

else

{

int[] day\_int = new int[day\_words.Length];

if (!int.TryParse(day\_words[0], out day\_int[0]))

{

throw new IncorrectSearchDataException();

}

if (!int.TryParse(day\_words[1], out day\_int[1]))

{

throw new IncorrectSearchDataException();

}

if (!int.TryParse(day\_words[2], out day\_int[2]))

{

throw new IncorrectSearchDataException();

}

if ((day\_int[1] > 12) || (day\_int[1] < 1))

{

throw new IncorrectSearchDataException();

}

if (day\_int[2] < 1)

{

throw new IncorrectSearchDataException();

}

if (((day\_int[1] == 2) && (day\_int[2] % 4 != 0) && (day\_int[0] > 28)) || ((day\_int[1] == 2) && (day\_int[2] % 4 == 0) && (day\_int[0] > 29)) || (((day\_int[1] == 1) || (day\_int[1] == 3) || (day\_int[1] == 5) || (day\_int[1] == 7) || (day\_int[1] == 8) || (day\_int[1] == 10) || (day\_int[1] == 12)) && day\_int[0] > 31) || (((day\_int[1] == 4) || (day\_int[1] == 6) || (day\_int[1] == 9) || (day\_int[1] == 11)) && day\_int[0] > 30) || day\_int[0] < 1)

{

throw new IncorrectSearchDataException();

}

this.SearchData.Date.Day = day\_int[0];

this.SearchData.Date.Month = day\_int[1];

this.SearchData.Date.Year = day\_int[2];

}

this.searchOn = true;

this.filterOn = false;

this.UpdateDataGrid();

}

catch (IncorrectSearchDataException exept)

{

exept.ShowMessageBox();

}

}

catch (EmptyDataException exept)

{

exept.ShowMessageBox();

}

}

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

{

if (!this.addBoxOpen)

{

this.splitContainer4.Panel1.Enabled = true;

this.splitContainer4.Panel1Collapsed = false;

this.addBoxOpen = true;

this.addMenuItem.Image = Properties.Resources.add;

if (!this.sellBoxOpen)

{

this.splitContainer1.Panel1Collapsed = false;

this.splitContainer1.Panel1.Enabled = true;

this.splitContainer4.Panel2Collapsed = true;

this.splitContainer4.Panel2.Enabled = false;

}

}

else

{

this.splitContainer4.Panel1Collapsed = true;

this.splitContainer4.Panel1.Enabled = false;

this.addBoxOpen = false;

this.addMenuItem.Image = Properties.Resources.add\_closed;

if (!this.sellBoxOpen)

{

this.splitContainer1.Panel1Collapsed = true;

this.splitContainer1.Panel1.Enabled = false;

}

}

}

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

{

if (!this.searchBoxOpen)

{

this.splitContainer3.Panel2Collapsed = false;

this.splitContainer3.Panel2.Enabled = true;

this.searchBoxOpen = true;

this.searchMenuItem.Image = Properties.Resources.search;

if (!this.filterBoxOpen)

{

this.splitContainer2.Panel1Collapsed = false;

this.splitContainer2.Panel1.Enabled = true;

this.splitContainer3.Panel1Collapsed = true;

this.splitContainer3.Panel1.Enabled = false;

}

}

else

{

this.splitContainer3.Panel2Collapsed = true;

this.splitContainer3.Panel2.Enabled = false;

this.searchBoxOpen = false;

this.searchMenuItem.Image = Properties.Resources.search\_closed;

if (!this.filterBoxOpen)

{

this.splitContainer2.Panel1Collapsed = true;

this.splitContainer2.Panel1.Enabled = false;

}

}

}

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

{

if (!this.filterBoxOpen)

{

this.splitContainer3.Panel1Collapsed = false;

this.splitContainer3.Panel1.Enabled = true;

this.filterBoxOpen = true;

this.filterMenuItem.Image = Properties.Resources.adjust;

if (!this.searchBoxOpen)

{

this.splitContainer2.Panel1Collapsed = false;

this.splitContainer2.Panel1.Enabled = true;

this.splitContainer3.Panel2Collapsed = true;

this.splitContainer3.Panel2.Enabled = false;

}

}

else

{

this.splitContainer3.Panel1Collapsed = true;

this.splitContainer3.Panel1.Enabled = false;

this.filterBoxOpen = false;

this.filterMenuItem.Image = Properties.Resources.adjust\_closed;

if (!this.searchBoxOpen)

{

this.splitContainer2.Panel1Collapsed = true;

this.splitContainer2.Panel1.Enabled = false;

}

}

}

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

{

if (!this.sellBoxOpen)

{

this.splitContainer4.Panel2.Enabled = true;

this.splitContainer4.Panel2Collapsed = false;

this.sellBoxOpen = true;

this.sellMenuItem.Image = Properties.Resources.sell;

if (!this.addBoxOpen)

{

this.splitContainer1.Panel1Collapsed = false;

this.splitContainer1.Panel1.Enabled = true;

this.splitContainer4.Panel1Collapsed = true;

this.splitContainer4.Panel1.Enabled = false;

}

}

else

{

this.splitContainer4.Panel2Collapsed = true;

this.splitContainer4.Panel2.Enabled = false;

this.sellBoxOpen = false;

this.sellMenuItem.Image = Properties.Resources.sell\_closed;

if (!this.addBoxOpen)

{

this.splitContainer1.Panel1Collapsed = true;

this.splitContainer1.Panel1.Enabled = false;

}

}

}

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

{

string path = "";

using (SaveFileDialog saveFileDialog = new SaveFileDialog())

{

saveFileDialog.InitialDirectory = "c:\\";

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

saveFileDialog.FilterIndex = 2;

saveFileDialog.RestoreDirectory = true;

saveFileDialog.OverwritePrompt = false;

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

if(saveFileDialog.FileName != "")

{

path = saveFileDialog.FileName;

}

}

}

this.products.Save\_file(path);

}

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

{

string path = "";

using (SaveFileDialog saveFileDialog = new SaveFileDialog())

{

saveFileDialog.InitialDirectory = "c:\\";

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

saveFileDialog.FilterIndex = 2;

saveFileDialog.RestoreDirectory = true;

saveFileDialog.OverwritePrompt = false;

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

if (saveFileDialog.FileName != "")

{

path = saveFileDialog.FileName;

}

}

}

this.products.Add\_file(path);

}

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

{

string path = "";

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.InitialDirectory = "c:\\";

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

openFileDialog.FilterIndex = 2;

openFileDialog.RestoreDirectory = true;

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

if (!String.IsNullOrEmpty(openFileDialog.FileName))

path = openFileDialog.FileName;

}

}

this.products.Read\_file(path);

this.UpdateDataGrid();

}

private void UpdateDataGrid()

{

this.dataGridView1.Rows.Clear();

if (this.searchOn)

{

for(int i = 0; i < this.products.Data.Count; i++)

{

bool searched = true;

if (this.products.Data[i].Name != this.SearchData.Name && this.SearchData.Name != "")

{

searched = false;

}

if (this.products.Data[i].Unit != this.SearchData.Unit && this.SearchData.Unit != "")

{

searched = false;

}

if (this.products.Data[i].Price != this.SearchData.Price && this.SearchData.Price != 0)

{

searched = false;

}

if (this.products.Data[i].Amount != this.SearchData.Amount && this.SearchData.Amount != 0)

{

searched = false;

}

if (this.products.Data[i].Date.Day != this.SearchData.Date.Day && this.SearchData.Date.Day != 0)

{

searched = false;

}

if (this.products.Data[i].Date.Month != this.SearchData.Date.Month && this.SearchData.Date.Month != 0)

{

searched = false;

}

if (this.products.Data[i].Date.Year != this.SearchData.Date.Year && this.SearchData.Date.Year != 0)

{

searched = false;

}

if (searched)

{

this.Add\_product\_To\_DataGridView(this.products.Data[i]);

}

}

}

else if (this.filterOn)

{

for (int i = 0; i < this.products.Data.Count; i++)

{

bool searched = true;

if (this.products.Data[i].Name != this.FilterData.Name && this.FilterData.Name != "")

{

searched = false;

}

if (this.products.Data[i].Unit != this.FilterData.Unit && this.FilterData.Unit != "")

{

searched = false;

}

if (this.products.Data[i].Price != this.FilterData.Price && this.FilterData.Price != 0)

{

searched = false;

}

if (this.products.Data[i].Amount != this.FilterData.Amount && this.FilterData.Amount != 0)

{

searched = false;

}

if (this.products.Data[i].Date.Day != this.FilterData.Date.Day && this.FilterData.Date.Day != 0)

{

searched = false;

}

if (this.products.Data[i].Date.Month != this.FilterData.Date.Month && this.FilterData.Date.Month != 0)

{

searched = false;

}

if (this.products.Data[i].Date.Year != this.FilterData.Date.Year && this.FilterData.Date.Year != 0)

{

searched = false;

}

if (searched)

{

this.Add\_product\_To\_DataGridView(this.products.Data[i]);

}

}

}

else

{

for (int i = 0; i < this.products.Data.Count; i++)

{

this.Add\_product\_To\_DataGridView(this.products.Data[i]);

}

}

}

private void Add\_product\_To\_DataGridView(Product prod)

{

this.dataGridView1.Rows.Add(prod.Name, prod.Unit, prod.Price.ToString(), prod.Amount.ToString(), prod.Date.Day.ToString() + "." + prod.Date.Month.ToString() + "." + prod.Date.Year.ToString());

}

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

{

try

{

bool dataUnCorrect = false;

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

{

if (RegisterBoxes[i].Text == "")

{

dataUnCorrect = true;

break;

}

}

if (dataUnCorrect)

{

throw new IncorrectRegisterDataException();

}

else

{

try {

Product new\_prod = new Product();

new\_prod.Name = this.RegisterBoxes[0].Text;

new\_prod.Unit = this.RegisterBoxes[1].Text;

double register\_price = 0;

if (double.TryParse(this.RegisterBoxes[2].Text, out register\_price))

{

new\_prod.Price = register\_price;

}

else

{

throw new IncorrectRegisterDataException();

}

int register\_amount = 0;

if (int.TryParse(this.RegisterBoxes[3].Text, out register\_amount))

{

new\_prod.Amount = register\_amount;

}

else

{

throw new IncorrectRegisterDataException();

}

string[] day\_words = this.RegisterBoxes[4].Text.Split('.');

if (day\_words.Length != 3)

{

throw new IncorrectRegisterDataException();

}

else

{

int[] day\_int = new int[day\_words.Length];

if (!int.TryParse(day\_words[0], out day\_int[0]))

{

throw new IncorrectRegisterDataException();

}

if (!int.TryParse(day\_words[1], out day\_int[1]))

{

throw new IncorrectRegisterDataException();

}

if (!int.TryParse(day\_words[2], out day\_int[2]))

{

throw new IncorrectRegisterDataException();

}

if ((day\_int[1] > 12) || (day\_int[1] < 1))

{

throw new IncorrectRegisterDataException();

}

if (day\_int[2] < 1)

{

throw new IncorrectRegisterDataException();

}

if (((day\_int[1] == 2) && (day\_int[2]%4!=0) && (day\_int[0]>28))||((day\_int[1] == 2) && (day\_int[2] % 4 == 0) && (day\_int[0] > 29)) || (((day\_int[1] == 1)|| (day\_int[1] == 3)|| (day\_int[1] == 5)|| (day\_int[1] == 7)|| (day\_int[1] == 8)|| (day\_int[1] == 10)|| (day\_int[1] == 12)) && day\_int[0] > 31) || (((day\_int[1] == 4)|| (day\_int[1] == 6)|| (day\_int[1] == 9)|| (day\_int[1] == 11)) && day\_int[0] > 30) || day\_int[0] < 1)

{

throw new IncorrectRegisterDataException();

}

new\_prod.Date.Day = day\_int[0];

new\_prod.Date.Month = day\_int[1];

new\_prod.Date.Year = day\_int[2];

}

bool productsExist = false;

if (this.products != null)

{

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

{

if (this.products.Data[i].Name == new\_prod.Name && this.products.Data[i].Unit == new\_prod.Unit && this.products.Data[i].Price == new\_prod.Price && this.products.Data[i].Date.Day == new\_prod.Date.Day && this.products.Data[i].Date.Month == new\_prod.Date.Month && this.products.Data[i].Date.Year == new\_prod.Date.Year)

{

this.products.Data[i].Amount += new\_prod.Amount;

productsExist = true;

break;

}

}

}

if (!productsExist)

{

this.products.Add\_product(new\_prod);

}

this.UpdateDataGrid();

}

catch(IncorrectRegisterDataException exept)

{

exept.ShowMessageBox();

}

}

}

catch (EmptyDataException exept)

{

exept.ShowMessageBox();

}

}

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

{

this.searchOn = false;

this.filterOn = false;

this.UpdateDataGrid();

}

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

{

bool dataEmpty = true;

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

{

if (FilterBoxes[i].Text != "")

{

dataEmpty = false;

break;

}

}

if (dataEmpty)

{

this.searchOn = false;

this.filterOn = false;

this.UpdateDataGrid();

return;

}

try

{

this.FilterData.Name = this.FilterBoxes[0].Text;

this.FilterData.Unit = this.FilterBoxes[1].Text;

double filter\_price = 0;

if (double.TryParse(this.FilterBoxes[2].Text, out filter\_price))

{

this.FilterData.Price = filter\_price;

}

int filter\_amount = 0;

if (int.TryParse(this.FilterBoxes[3].Text, out filter\_amount))

{

this.FilterData.Amount = filter\_amount;

}

string[] day\_words = this.FilterBoxes[4].Text.Split('.');

if (day\_words.Length == 3 || (day\_words.Length == 1 && day\_words[0] == ""))

{

if (day\_words.Length == 3)

{

int[] day\_int = new int[day\_words.Length];

if (!int.TryParse(day\_words[0], out day\_int[0]))

{

throw new IncorrectFilterDataException();

}

if (!int.TryParse(day\_words[1], out day\_int[1]))

{

throw new IncorrectFilterDataException();

}

if (!int.TryParse(day\_words[2], out day\_int[2]))

{

throw new IncorrectFilterDataException();

}

if ((day\_int[1] > 12) || (day\_int[1] < 1))

{

throw new IncorrectFilterDataException();

}

if (day\_int[2] < 1)

{

throw new IncorrectFilterDataException();

}

if (((day\_int[1] == 2) && (day\_int[2] % 4 != 0) && (day\_int[0] > 28)) || ((day\_int[1] == 2) && (day\_int[2] % 4 == 0) && (day\_int[0] > 29)) || (((day\_int[1] == 1) || (day\_int[1] == 3) || (day\_int[1] == 5) || (day\_int[1] == 7) || (day\_int[1] == 8) || (day\_int[1] == 10) || (day\_int[1] == 12)) && day\_int[0] > 31) || (((day\_int[1] == 4) || (day\_int[1] == 6) || (day\_int[1] == 9) || (day\_int[1] == 11)) && day\_int[0] > 30) || day\_int[0] < 1)

{

throw new IncorrectFilterDataException();

}

this.FilterData.Date.Day = day\_int[0];

this.FilterData.Date.Month = day\_int[1];

this.FilterData.Date.Year = day\_int[2];

}

}

else

{

throw new IncorrectFilterDataException();

}

this.searchOn = false;

this.filterOn = true;

this.UpdateDataGrid();

}

catch (IncorrectFilterDataException exept)

{

exept.ShowMessageBox();

}

}

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

{

try

{

bool dataEmpty = false;

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

{

if (SearchBoxes[i].Text == "")

{

dataEmpty = true;

break;

}

}

if (dataEmpty)

{

throw new EmptySearchDataException();

}

try

{

this.SearchData.Name = this.SearchBoxes[0].Text;

this.SearchData.Unit = this.SearchBoxes[1].Text;

double search\_price = 0;

if (double.TryParse(this.SearchBoxes[2].Text, out search\_price))

{

this.SearchData.Price = search\_price;

}

else

{

throw new IncorrectWriteOffDataException();

}

int search\_amount = 0;

if (int.TryParse(this.SearchBoxes[3].Text, out search\_amount))

{

this.SearchData.Amount = search\_amount;

}

else

{

throw new IncorrectWriteOffDataException();

}

string[] day\_words = this.SearchBoxes[4].Text.Split('.');

if (day\_words.Length != 3)

{

throw new IncorrectWriteOffDataException();

}

else

{

int[] day\_int = new int[day\_words.Length];

if (!int.TryParse(day\_words[0], out day\_int[0]))

{

throw new IncorrectWriteOffDataException();

}

if (!int.TryParse(day\_words[1], out day\_int[1]))

{

throw new IncorrectWriteOffDataException();

}

if (!int.TryParse(day\_words[2], out day\_int[2]))

{

throw new IncorrectWriteOffDataException();

}

if ((day\_int[1] > 12) || (day\_int[1] < 1))

{

throw new IncorrectWriteOffDataException();

}

if (day\_int[2] < 1)

{

throw new IncorrectWriteOffDataException();

}

if (((day\_int[1] == 2) && (day\_int[2] % 4 != 0) && (day\_int[0] > 28)) || ((day\_int[1] == 2) && (day\_int[2] % 4 == 0) && (day\_int[0] > 29)) || (((day\_int[1] == 1) || (day\_int[1] == 3) || (day\_int[1] == 5) || (day\_int[1] == 7) || (day\_int[1] == 8) || (day\_int[1] == 10) || (day\_int[1] == 12)) && day\_int[0] > 31) || (((day\_int[1] == 4) || (day\_int[1] == 6) || (day\_int[1] == 9) || (day\_int[1] == 11)) && day\_int[0] > 30) || day\_int[0] < 1)

{

throw new IncorrectWriteOffDataException();

}

this.SearchData.Date.Day = day\_int[0];

this.SearchData.Date.Month = day\_int[1];

this.SearchData.Date.Year = day\_int[2];

}

bool exist\_product = false;

for(int i = 0; i < this.products.Data.Count; i++)

{

if (this.products.Data[i].Name == this.SearchData.Name&& this.products.Data[i].Unit == this.SearchData.Unit && this.products.Data[i].Price == this.SearchData.Price && this.products.Data[i].Date.Day == this.SearchData.Date.Day && this.products.Data[i].Date.Month == this.SearchData.Date.Month && this.products.Data[i].Date.Year == this.SearchData.Date.Year)

{

if (this.products.Data[i].Amount >= this.SearchData.Amount)

{

this.products.Data[i].Amount -= this.SearchData.Amount;

if(this.products.Data[i].Amount <= 0)

{

this.products.Data.Remove(this.products.Data[i]);

}

exist\_product = true;

}

else

{

throw new NotEnoughAmount();

}

}

}

if (!exist\_product)

{

throw new IncorrectWriteOffDataException();

}

this.UpdateDataGrid();

}

catch (IncorrectWriteOffDataException exept)

{

exept.ShowMessageBox();

}

catch(NotEnoughAmount exept)

{

exept.ShowMessageBox();

}

}

catch (EmptyDataException exept)

{

exept.ShowMessageBox();

}

}

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

{

double sum = 0;

for(int i = 0; i < this.products.Data.Count; i++)

{

sum += this.products.Data[i].Price \* this.products.Data[i].Amount;

}

MessageBox.Show("Sum: "+sum.ToString(), "Product sum", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

}

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

{

try

{

this.SellData = new Product();

bool dataEmpty = false;

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

{

if (SellBoxes[i].Text == "")

{

dataEmpty = true;

break;

}

}

if (dataEmpty)

{

throw new IncorrectSellDataException();

}

this.SellData.Name = this.SellBoxes[0].Text;

this.SellData.Unit = this.SellBoxes[1].Text;

double sell\_price = 0;

if (double.TryParse(this.SellBoxes[2].Text, out sell\_price))

{

this.SellData.Price = sell\_price;

}

else

{

throw new IncorrectSellDataException();

}

int sell\_amount = 0;

if (int.TryParse(this.SellBoxes[3].Text, out sell\_amount))

{

this.SellData.Amount = sell\_amount;

}

else

{

throw new IncorrectSellDataException();

}

int sell\_all\_amount = this.SellData.Amount;

bool exist = false;

for (int i = 0; i < this.basket.Data.Count; i++)

{

if (this.basket.Data[i].Name == this.SellData.Name && this.basket.Data[i].Unit == this.SellData.Unit && this.basket.Data[i].Price == this.SellData.Price)

{

exist = true;

sell\_all\_amount += this.basket.Data[i].Amount;

}

}

bool exist\_product = false;

int product\_all\_amount = 0;

for (int i = 0; i < this.products.Data.Count; i++)

{

if (this.products.Data[i].Name == this.SellData.Name && this.products.Data[i].Unit == this.SellData.Unit && this.products.Data[i].Price == this.SellData.Price)

{

product\_all\_amount += this.products.Data[i].Amount;

exist\_product = true;

}

}

if (!exist\_product)

{

throw new IncorrectSellDataException();

}

if (product\_all\_amount < sell\_all\_amount)

{

throw new NotEnoughAmount();

}

else

{

if (exist)

{

for (int i = 0; i < this.basket.Data.Count; i++)

{

if (this.basket.Data[i].Name == this.SellData.Name && this.basket.Data[i].Unit == this.SellData.Unit && this.basket.Data[i].Price == this.SellData.Price)

{

this.basket.Data[i].Amount += this.SellData.Amount;

break;

}

}

}

else

{

this.basket.Add\_product(this.SellData);

}

}

if (this.Basket\_window != null)

{

this.Basket\_window.updateBasketDataGrid(this.basket);

}

}

catch (NotEnoughAmount exept)

{

exept.ShowMessageBox();

}

catch (IncorrectSellDataException exept)

{

exept.ShowMessageBox();

}

}

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

{

try

{

string bill\_text = "";

double sum = 0;

for (int i = 0; i < this.basket.Data.Count; i++)

{

int current\_amount = this.basket.Data[i].Amount;

for(int j = 0; j < this.products.Data.Count; j++)

{

if (this.products.Data[j].Name == this.basket.Data[i].Name && this.products.Data[j].Unit == this.basket.Data[i].Unit && this.products.Data[j].Price == this.basket.Data[i].Price)

{

if (this.products.Data[j].Amount < current\_amount)

{

current\_amount -= this.products.Data[j].Amount;

this.products.Data[j].Amount = 0;

this.products.Data.Remove(this.products.Data[j]);

j--;

}

else if(this.products.Data[j].Amount == current\_amount)

{

current\_amount -= this.products.Data[j].Amount;

this.products.Data[j].Amount = 0;

this.products.Data.Remove(this.products.Data[j]);

break;

}

else

{

this.products.Data[j].Amount -= current\_amount;

current\_amount = 0;

break;

}

}

}

if(current\_amount > 0)

{

throw new NotEnoughAmount();

}

else

{

sum += this.basket.Data[i].Price \* this.basket.Data[i].Amount;

bill\_text += this.basket.Data[i].Name + "\t" + this.basket.Data[i].Unit + "\t" + this.basket.Data[i].Price.ToString() + "\t" + this.basket.Data[i].Amount.ToString() + "\t" + (this.basket.Data[i].Price \* this.basket.Data[i].Amount).ToString()+"\n";

}

}

this.basket.Data.Clear();

if (this.Basket\_window != null)

{

this.Basket\_window.updateBasketDataGrid(this.basket);

}

this.UpdateDataGrid();

this.bill = new Bill(bill\_text, sum);

this.bill.Show();

}

catch (NotEnoughAmount exept)

{

exept.ShowMessageBox();

}

}

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

{

if (!this.basketOpen)

{

this.Basket\_window = new Basket(this);

this.Basket\_window.Show();

this.Basket\_window.updateBasketDataGrid(this.basket);

this.basketMenuItem.Image = Properties.Resources.basket;

this.basketOpen = true;

}

else

{

if(this.Basket\_window != null)

{

this.Basket\_window.Close();

this.basketMenuItem.Image = Properties.Resources.basket\_closed;

this.basketOpen = false;

}

}

}

public void basket\_closing()

{

this.basketMenuItem.Image = Properties.Resources.basket\_closed;

}

public void about\_us\_closing()

{

this.aboutUsMenuItem.Image = Properties.Resources.about\_us\_closed;

this.aboutOpen = false;

}

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

{

if (!this.aboutOpen)

{

this.aboutUsMenuItem.Image = Properties.Resources.about\_us;

this.aboutUs = new AboutUs(this);

this.aboutUs.Show();

this.aboutOpen = true;

}

else

{

this.aboutUsMenuItem.Image = Properties.Resources.about\_us\_closed;

if(this.aboutUs != null)

{

this.aboutUs.Close();

}

this.aboutOpen = false;

}

}

public void help\_closing()

{

this.helpMenuItem.Image = Properties.Resources.help\_closed;

}

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

{

if (!this.helpOpen)

{

this.helpMenuItem.Image = Properties.Resources.help;

this.help = new Help(this);

this.help.Show();

this.helpOpen = true;

}

else

{

this.helpOpen = false;

this.helpMenuItem.Image = Properties.Resources.help\_closed;

if(this.help != null)

{

this.help.Close();

}

}

}

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

{

this.products.Data.Clear();

this.basket.Data.Clear();

this.UpdateDataGrid();

if(this.Basket\_window!= null)

{

this.Basket\_window.updateBasketDataGrid(this.basket);

}

this.searchOn = false;

this.filterOn = false;

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Thesis

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.SetHighDpiMode(HighDpiMode.SystemAware);

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}