МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЖИТОМИРСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Кафедра інженерії програмного забезпечення

**КУРСОВИЙ ПРОЕКТ**

(ПОЯСНЮВАЛЬНА ЗАПИСКА)

з дисципліни: «Об’єктно-орієнтоване програмування»

на тему:

**«Інформаційно-пошукова система: структура університету»**

студентки І курсу групи ПІ-55

спеціальності 121 «Інженерія програмного забезпечення»

Луцевич Олександри Олександрівни

(прізвище, ім’я та по-батькові)

Керівник: доцент кафедри КІ Морозов А. В.

Дата захисту: " \_\_\_ " \_\_\_\_\_\_\_\_\_\_\_\_ 2017 р.

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

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Морозов \_\_\_\_\_

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Л. Левківський .

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.В. Марчук \_\_\_\_\_\_

(підпис) (прізвище та ініціали)

Житомир – 2017

ЖИТОМИРСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційно-комп’ютерних технологій

Кафедра інженерії програмного забезпечення

Освітній рівень: бакалавр

Спеціальність 121 «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»

Зав. кафедри

\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Панішев

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_2017р.

ЗАВДАННЯ

НА КУРСОВИЙ ПРОЕКТ СТУДЕНТУ

Луцевич Олександрі Олександрівні

1. Тема роботи: Розробка інформаційно-пошукової системи: структура університету

керівник роботи: Морозов Андрій Васильович.

1. Строк подання студентом: “\_ ”\_червня\_\_2017р.
2. Вихідні дані до роботи: Розробити інформаційно-пошукову систему: структура університету,
3. Зміст розрахунково-пояснювальної записки(перелік питань. Які підлягають розробці)
   * + 1. Технічне завдання
       2. Аналіз аналогічних розробок
       3. Алгоритми роботи програми

4. Опис роботи програми

5. Програмне дослідження

1. Перелік графічного матеріалу(з точним зазначенням обов’язкових креслень)
   * + 1. Презентація до курсового проекту;
       2. Посилання на репозиторій;
2. Консультанти розділів проекту (роботи)

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посади консультанта | Підпис, дата | |
| завдання  видав | завдання прийняв |
| 1,2 | Левківський В.Л., ст. викладач каф. ІПЗ |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. Дата видачі завдання “\_04\_”\_\_квітня\_\_2017 р.

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів курсової роботи | Строк виконання етапів роботи | Примітки |
| 1 | Постановка задачі | 04.04.17 -05.04.17 |  |
| 2 | Пошук, огляд та аналіз аналогічних розробок | 06.04.17 -15.04.17 |  |
| 3 | Формулювання технічного завдання | 16.04.17 – 17.04.17 |  |
| 4 | Опрацювання літературних джерел | 18.04.17 – 28.04.17 |  |
| 5 | Проектування структури | 29.04.17 – 10.05.17 |  |
| 6 | Написання програмного коду | 11.05.17 – 18.06.17 |  |
| 7 | Налагодження | 19.06.17 – 20.06.17 |  |
| 8 | Написання пояснювальної записки | 20.06.17 –  21.06.17 |  |
| 9 | Захист | 22.06.17 |  |

**КАЛЕНДАРНИЙ ПЛАН**

**Студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Луцевич О. О.

(підпис) (прізвище та ініціали)

**Керівник роботи** \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Морозов А. В.

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до курсового проекту на тему «Інформаційно – пошукова система “ структура університету”» складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаної літератури та додатку.

Текстова частина викладена на 33 сторінках друкованого тексту.

Пояснювальна записка має 66 сторінку додатків. Список використаних джерел містить 5 найменувань і займає 1 сторінку. В роботі наведено 17 рисунків. Загальний обсяг роботи – 102 сторінки.

У першому розділі було обґрунтовано створення програми для зберігання, пошуку і відображення інформації про структуру університету.

У другому розділі проведено проектування і розробка програмного продукту.

У третьому розділі проведено тестування програмного продукту.

Висновок містить в собі результати виконаної роботи створення програми для зберігання, пошуку і відображення інформації про країни.

У додатку представлений лістинг розробленого програмного продукту.

Ключові слова: ООП, БАЗА ДАНИХ, ВІДОБРАЖЕННЯ ІФОРМАЦІЇ, ДОСТУП, ЗБЕРЕЖЕННЯ ІНФОРМАЦІЇ, ПОШУК, ВИКЛАДАЧ, СТУДЕНТ, ГРУПА СТУДЕНТІВ, КЛАС.

ЗМІСТ

[ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 6](#_Toc485861458)

[ВСТУП 7](#_Toc485861459)

[РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ 9](#_Toc485861460)

[1.1 Аналіз задачі, засобів та методів її вирішення 9](#_Toc485861461)

[1.2 Аналіз існуючого програмного забезпечення за тематикою курсової роботи. 11](#_Toc485861462)

[1.3 Технічне завдання на курсову роботу 13](#_Toc485861463)

[Висновки з першого розділу 14](#_Toc485861464)

[РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 15](#_Toc485861465)

[2.1 Проектування загального алгоритму роботи програми 15](#_Toc485861466)

[2.3 Розробка програмного забезпечення 19](#_Toc485861467)

[РОЗДІЛ 3 ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ 27](#_Toc485861468)

[3.1 Опис роботи додатку 27](#_Toc485861469)

[ВИСНОВКИ 34](#_Toc485861470)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 35](#_Toc485861471)

[ДОДАТКИ 36](#_Toc485861472)

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

VS – Visual Studio

ПЗ – Програмне забезпечення

ІПС– Інформаційно-пошукова система

ІПМ– Інформаційно-пошукова мова

БД –База даних

# ВСТУП

У цій курсовій роботі буде наведено процес створення інформаційно-пошукової системи: структура університету.

Інформаційно–пошукові системи — це різновид автоматизованих інформаційних систем, в яких завершальна обробка даних не передбачається. Ці системи призначені для пошуку текстів (документів, їх частин, фактографічних записів) в сховищах (базах даних) за формальними характеристиками. Тому в роботі ІПС можна виділити два основних етапи: перший — збір і зберігання інформації, другий — пошук і видача інформації користувачам.

В економіці ІПС можуть використовуватися для зберігання і пошуку нормативних, планових, бухгалтерських та інших документів, даних для наукових досліджень.

Інформаційно-пошукові системи з'явилися досить давно. Теорії і практиці побудови таких систем присвячена досить велика кількість статей, основна маса яких приходиться на кінець 70-х - початок 80-х років. Тобто не можна сказати, що з появою Інтернету і бурхливим входженням його в практику інформаційного забезпечення, з'явилося щось принципово нове, чого не було раніше. І на сьогоднішній день немає іншого способу швидкого пошуку даних, крім пошуку за ключовими словами.

Абонент звертається до ІПС з інформаційним запитом - текстом, що відбиває інформаційну потребу даного абонента, наприклад, його бажання знайти список книг по теорії інформаційного пошуку чи список аптек, у яких можна купити потрібні ліки. Пошук інформації ведеться в пошуковому масиві, що формується (і в міру необхідності обновляється) розробниками чи адміністраторами системи. Елементи пошукового масиву вводяться в інформаційно-пошукову систему на природному (чи близькому до нього) мові, а потім звичайно піддаються індексуванню, тобто перекладу на формальну інформаційно-пошукову мову.

Актуальність роботи полягає в застосуванні сучасних технологій об'єктно-орієнтованого підходу в програмуванні. Проект зберігає та обробляє данні за участю реальних об'єктів як статичну модель реалізовану в програмному коді.

Мета роботи полягає в розробці програмного продукту - розробка програмного комплексу з використанням об'єктно-орієнтованого підходу і користувальницьких класів. Для досягнення мети необхідно провести об'єктно-орієнтований аналіз, об'єктно-орієнтоване проектування, об'єктно-орієнтоване програмування і зібрати статистичні результати.

Об'єкт дослідження курсової роботи - процес введення, обробки, зберігання та відображення інформації про структуру університету.

Предмет дослідження – програма для обробки, зберігання та відображення інформації про структуру університету: методи збереження інформації в базі даних, методи обробки і перевірки даних.

ІПС це можливість збереження, контролю, аналізу та обробки отриманих даних. Це зручний спосіб для представлення великої кількості однотипної інформації.

# РОЗДІЛ 1 АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

## Аналіз задачі, засобів та методів її вирішення

Задача полягає в тому, що потрібно:

1. Визначити потреби звичайного користувача та Адміністратора
2. Зрозуміти потрібний об’єм інформації для користувача
3. Налагодити можливість пошуку по певним критеріям
4. Розмежувати запис викладача та студента
5. Зрозуміти яку колекцію краще використовувати для збереження даних
6. Розробити індивідуальний ключ для контролю записів в БД

В процесі створення БД брався будь-який аспект проекту, розроблялася його найлегша версія, а після того вдосконалювалася, чи повністю перероблювалася.

Було вирішено розробляти БД в Visual Studio за допомогою додатку Windows Presentation Foundation, оскільки він має ширші можливості для розробки інтерфейсу.

Спочатку було вирішено зберігати частину даних за допомогою колекції List. Але після розробки бібліотеки класів, яка включала в себе клас Студент та клас Викладач довелось розмежувати ці два типи та досягти ідентичного ключа для доступу та ідентифікації цих записів.

Тоді було вирішено розробити записи на основі колекції Dictionary. Це дозволить водночас і розрізняти різні класи і співставляти певні відомості між різними записами.

Також будо створено бібліотеку класів, яка дозволяє не повторювати схожі поля в класах викладач та студент.

Використовувалась робота з файлами для попереднього збереження та аналізу даних. Кінцеві дані як для звичайного користувача так і для адміністратора виводяться в файл. Таким чином ми надаємо інформацію не порушуючи структури самого файлу та надаючи доступ туди лише для адміністратора.

Тобто, звичайний користувач не бачить всієї тієї інформації, що доступна адміністратору. Адміністратору доступні такі додаткові поля як:

* дата народження
* ідентифікаційний код
* середній бал студента
* посада викладача

Тож, БД матиме вигляд декількох різних вікон для реалізації запланованого. Адміністратор повинен буде коректно ввести пароль для доступу до повного об’єму даних та їх зміни. Адміністратор має стандартні можливості плюс додаткові. Користувач – стандартні.

## 1.2 Аналіз існуючого програмного забезпечення за тематикою курсової роботи.

При аналізі вже існуючого ПЗ за тематикою курсової роботи було виявлено декілька проектів. Всі вони схожі один на одного і мають майже однакову реалізацію.

Всі знайдені зразки виконувались за допомогою SQL Server-a чи Access.

Всі вони мають схожу структуру та зв’язані між собою поля. Але загальна структура дозволяла зрозуміти, що частіше за все користувачу доступні всі поля, що відображаються адміністратором для нього. Наприклад рисунок 1.1, на ньому ми бачимо базу даних для інформації про студентів.

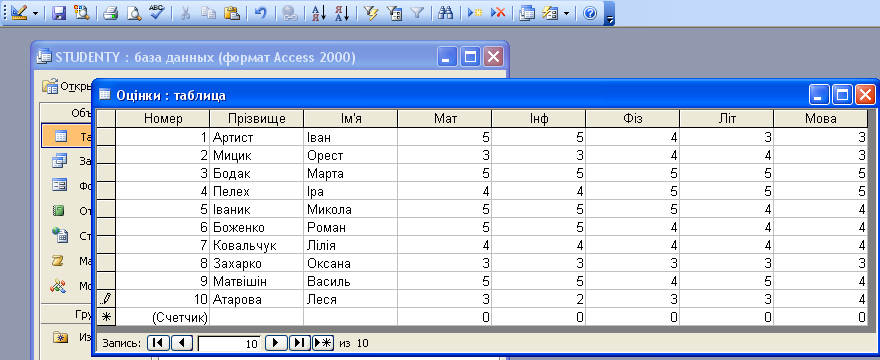


Рисунок 1.1 - Приклад БД розробленої в Access

На рисунку 1.2 як приклад зображено зв’язані поля та ідентифікація студентів серед списку за допомогою унікального неповторюваного і простого в зчитуванні та використанні ідентифікаційного коду. Це дає можливість контролювати нові та вже створені записи та не дозволяти копіюванню даних та складним доступом через однаковий ключ доступу.

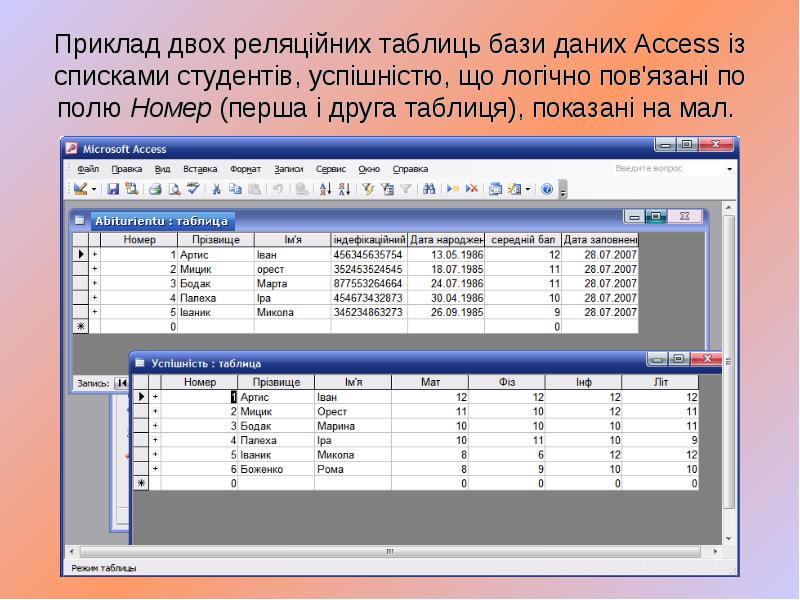


Рисунок 1.2 – Приклад БД з ключем ідентифікаційним кодом

В першому прикладі допускалась можливість створення однакових записів. Відповідно з другому прикладі така можливість заборонена, бо навіть особи-близнюки народжені в один день з одним і тим самим іменем будуть мати різний ідентифікаційний код, в якому зашифрована статі людини день народження та номер людини, що народжена в цей день та зареєстрована. Таким чином контроль даних за цим ключем є ідеальним варіантом для розгляду унікальності записів.

## 1.3 Технічне завдання на курсову роботу

**Підстава для розробки:**

Підставою для розробки курсової роботи є завдання на курсову роботу.

**Призначення розробки:**

Програма призначення для індивідуального використання користувачем чи адміністратором для різного допуску до великих об’ємів інформації про структуру університету, а саме викладацький склад та структура навчальних груп.

Вимоги до програмного засобу

БД повинна надати інформацію знайдену за ключовими словами та функціями. У відповідь на некоректні дії користувача повинно буде з’явитись попередження та надаватись коректна відповідь та підказки разом з можливістю знову виконати ті ж самі дії вже коректним методом.

Програма повинна виконувати наступні функції:

1. Виведення всієї інформації без коду та дат народження.
2. Сортування в окремі списки студентів та викладачів.
3. Пошук за заданими та поширеними критеріями.
4. Для адміністратора коректне додавання, вилучення та зміна записів з повним доступом до інформації.

Потрібно дотримуватись максимальної простоти в інтерфейсі і зрозумілості використання БД, як для звичайного користувача так і для адміністратора.

У разі виконання некоректних дій передбачити підказки та приклади, максимальне роз’яснення причини через яку певна дія не вдалась чи вдалась некоректно.

**Вимоги до надійності:**

Передбачити дії при виконанні некоректних дій як зі сторони звичайного користувача так і персону з правом доступу адміністратора при використанні продукту.

**Вимоги до складу і параметрів технічних засобів:**

Система повинна працювати на IBM сумісних комп’ютерах.

**Вимоги до інтерфейсу:**

Інтерфейс головного вікна додатку повинен виконувати функції входу до зони адміністратора з паролем та звичайного користувача без паролю.

**Вимоги до інформаційної і програмної сумісності:**

Повинна бути встановлена Visual Studio 2013 і вище.

**Спеціальні вимоги**

Немає

## Висновки з першого розділу

У ході виконання першого розділу було проаналізовано БД схожого зразка та вирішено питання доступу та ідентифікації.

Було визначено дозволені можливості обох типів користувачів та їх зону роботи, можливість доступу до інформації з різними рівнями безпеки.

Далі було визначено необхідні інструменти для реалізації програмного забезпечення, а також інтерфейсу та функціонування БД, що створюється.

# РОЗДІЛ 2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 2.1 Проектування загального алгоритму роботи програми

На даному етапі розробки розглянемо загальний алгоритм роботи БД.

Загальна схема роботи БД приведена на рисунку 2.1.

Опис загального алгоритму ПЗ наведено нижче залежно від можливостей доступу.

При запуску програми користувач натискає кнопку «Користувач» чи «Адмін». Під кнопками є коректне пояснення всіх можливостей, що має той чи інший користувач.

Якщо він обирає форму «Адмін» йому слід коректно ввести пароль, якщо користувач забув пароль, але в принципі знає підказка допоможе згадати пароль. Якщо пароль незручний його можна змінити ввівши старий та новий паролі. Після цього йому доступні такі можливості як «Додавання нового запису», «Видалення старого запису», «Зміна існуючого запису» та «Перегляд всіх даних».

У розділах Додавання та Зміна він обирає категорію: викладач чи студент, при створенні нових записів є можливість додавати групу студентів з відповідними ідентичними даними (назва групи, курс, факультет), ці дані потрібно ввести лише один раз, відповідно змінюючи лише особисті дані та додаючи потрібну кількість елементів. Видалення проводиться за допомогою введення ідентифікаційного коду особи, саме тому доступ до такої можливості надається лише адміністратору.

Кнопка Перегляд дозволяє продивитись інформацію про всіх наявних персон у БД.

Також при некоректному введенні даних з’являються додаткові підказки зі зразком введених даних. Регулярні вирази перевіряють коректність введених даних.

Якщо користувач обрав форму «Користувач» йому доступні три кнопки «Сортування», «Перегляд», «Пошук».

Сортування можливе за критеріями лише студенти лише викладачі. Пошук відрізняється полями для студента та викладача. Перегляд це можливість побачити всіх хто знаходиться в БД без ідентифікаційного коду, середнього балу для студента та посади, дня народження для викладача.

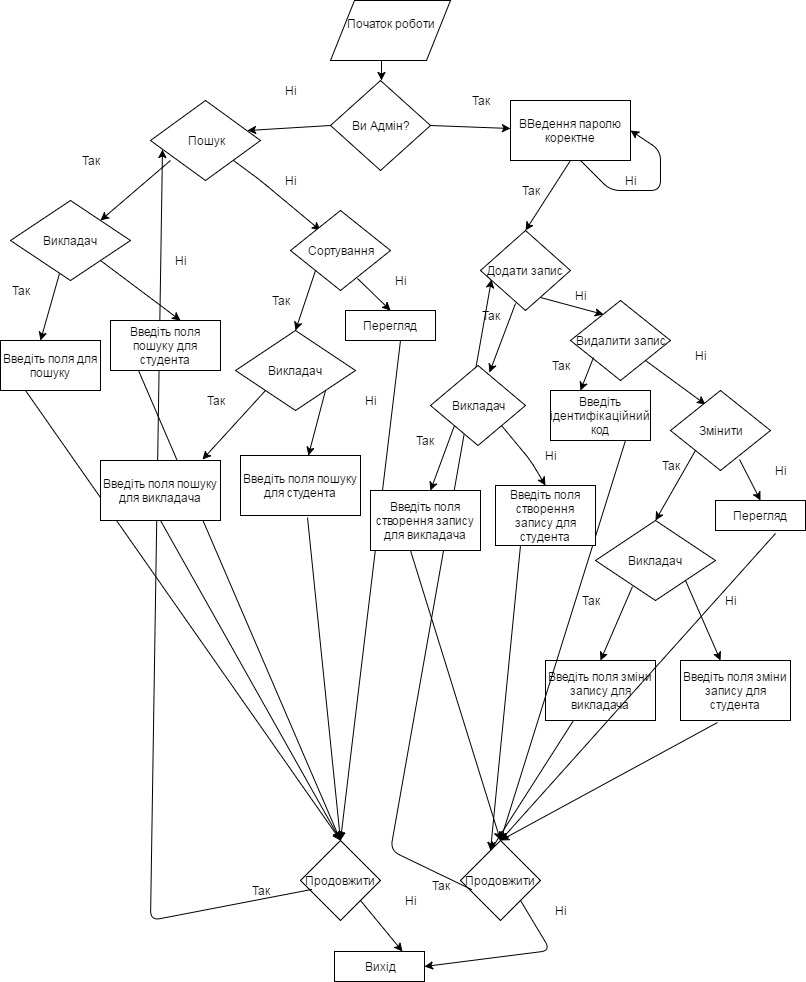


Рисунок 2.1 – Загальна схема роботи програми

2.2 Розробка функціональних алгоритмів роботи програми

Для розробки цієї БД використовувались малі за об’ємом та швидкі для виконання алгоритми зчитування та приведення до потрібного вигляду даних, таким чином певні дані були залишені в строковому вигляді, в той час як інші були приведені в різні цифрові типи даних. Введені дані аналізувались та розглядалась та чи інша вітка коду при частій перевірці того з яким класом ми будемо працювати, який об’єкт потрібно створити та на які помилки слід перевіряти ту чи іншу інформацію. Оскільки як для викладача так і для студента залишалася відкритою та ж сама форма ми мали аналізувати зроблений користувачем вибір та корегувати видимість та доступ до тих чи інших полів.

Для зчитування інформації з файлу використовувався простий, але дієвий алгоритм. Спочатку перевірявся весь файл відповідно до категорії чи існував вже такий запис раніше і виводиться повідомлення, що такий запис існує і його можна видалити, перезаписати, але неможливо додати інший запис з тим самим кодом. Якщо шукається інформація з певним кодом, то спочатку зчитується певна кількість полів, для викладача – 19, для студента – 21. Потім звіряється код (в формі для звичайного користувача додатково порівнюються ті поля, що були введені користувачем) і видається результат порівняння. Запис «Викладач» чи «Студент» виділяються великими літерами для більш зручного орієнтування. Для користувача, який користується такою формою рідше ніж адміністратор, що заповнює ці записи додатково виділяється характеристика приналежності певного запису до того чи іншого класу табуляцією.

## 2.3 Розробка програмного забезпечення

2.3.1 Розробка інтерфейсу головного меню.

Для розробки інтерфейсу було використано Visual Studio 2013 та створено додаток Windows Presentation Foundation. Оскільки він дозволяє реалізувати візуальні можливості зрозумілі як для адміністратора так і для звичайного користувача. Оскільки потрібно використовувати самостійно створений на основі колекції словник, було розроблено базовий клас PersonaUniversity та похідні від нього – Student та Lecturer. Розробка показана на рисунку 2.2.

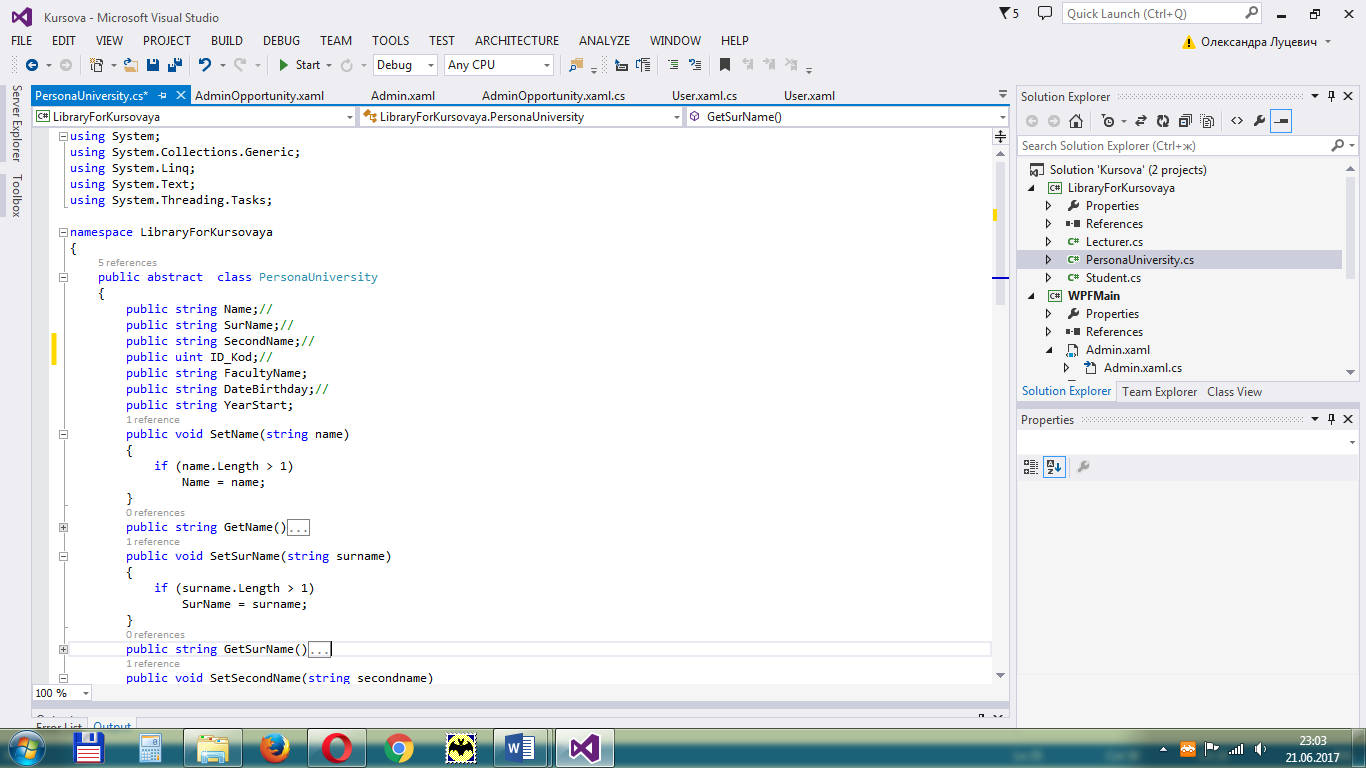


Рисунок 2.2 – Розробка класу PersonaUniversity

Після розробки класів було створено головну форму з якої починаються робота персони, яка скористається цією БД. Розробку зображено на рисунку 2.3.

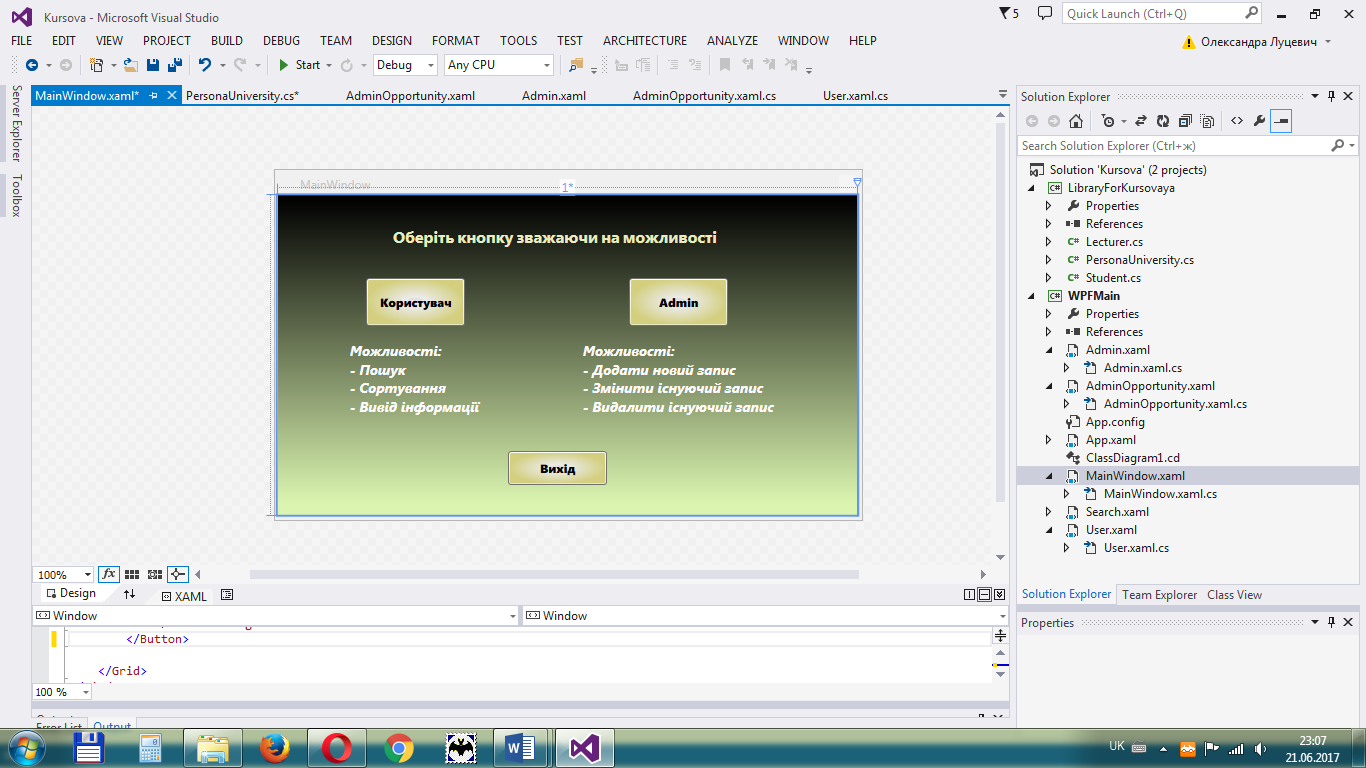


Рисунок 2.3 – Розробка головної першої форми

В пріоритеті для нас можливість допуску певної кількості користувачів до зони адміністратора, тому важливо зробити пароль невидимим та не копіюємим тому ми використовуємо спеціальний об’єкт PasswordBox. Для зміни пароля передусім потрібно ввести старий пароль та новий пароль, довжина якого буде не більшою за вісім символів та який також буде шифруватись зірочками. Цю можливість вказувати шифрування розглянуто на малюнку 2.4 у виділеному фрагменті.

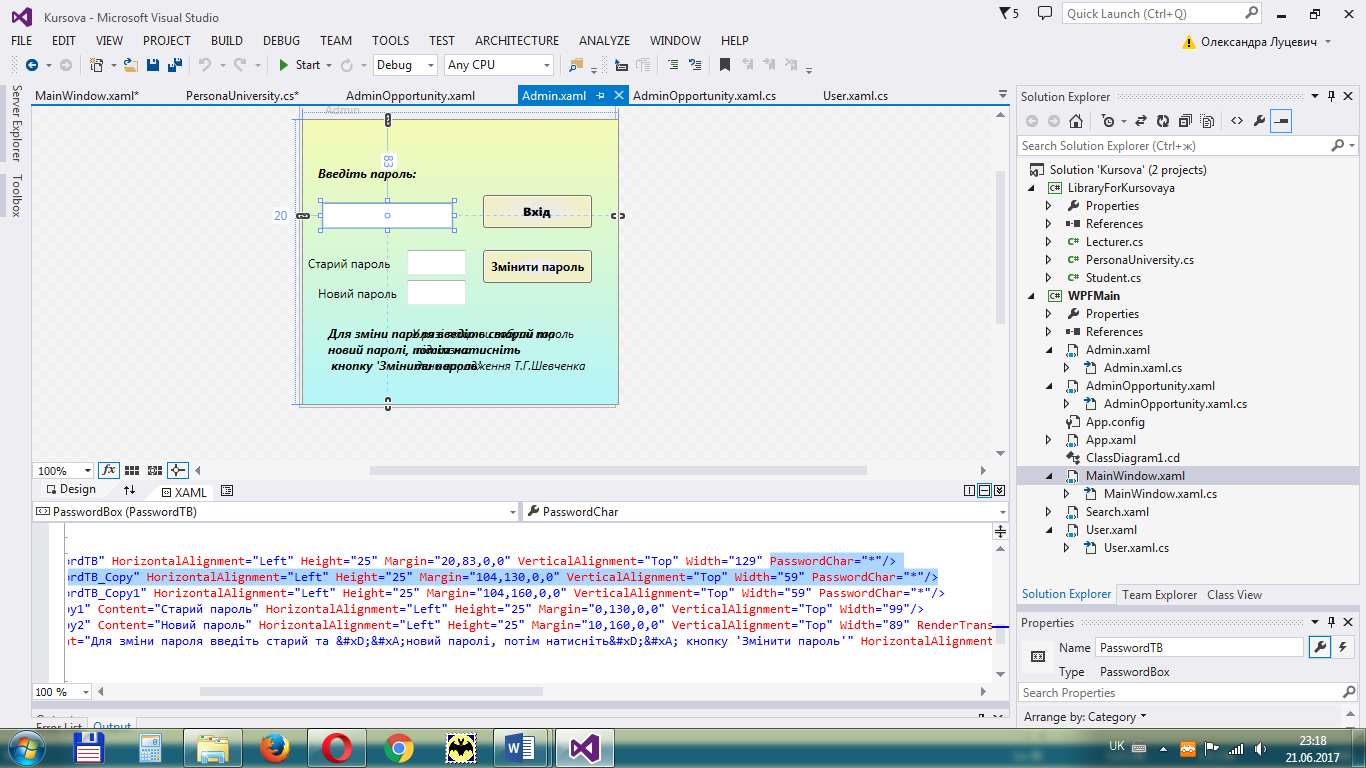


Рисунок 2.4 – Розробка з PasswordBox

Розглянемо код для зчитування звичайного пароля, який за замовчуванням буде 1814:

private void PasswordButton\_Click(object sender, RoutedEventArgs e)

{

Password\_Copy1.Visibility = Visibility.Visible;

Password\_Copy2.Visibility = Visibility.Visible;

Password = "1814";

Help.Visibility = Visibility.Visible;

PasswordTB\_Copy.Visibility = Visibility.Visible;

PasswordTB\_Copy1.Visibility = Visibility.Visible;

string PasswordOne = PasswordTB\_Copy.Password;

string PasswordTwo = PasswordTB\_Copy1.Password;

if (PasswordOne == Password)

{

Password = PasswordTwo;

Label1.Content = "Ви змінили пароль!";

Oznaka = true;

Label1.Visibility = Visibility.Visible;

Help.Visibility = Visibility.Hidden;

AdminOpportunity frm = new AdminOpportunity();

frm.Show();

}

}

private void Enter\_Click(object sender, RoutedEventArgs e)

{

if (!Oznaka)

{ Password = "1814"; }

string copy = PasswordTB.Password;

if (copy== Password)

{

AdminOpportunity frm = new AdminOpportunity();

frm.Show();

}

else

if(!Oznaka)

Label1.Visibility = Visibility.Visible;

}

Також потрібно не зробити помилкою та зчитувати введений пароль як символи не цифри лише потім конвертувати за потребою, це забезпечить виконання програми в будь-якому випадку. При зміні пароля повинні ставати видимими та активними певні підказки та поля типу PasswordBox. Після зміни пароля з’являтиметься попередження- нагадування про вже змінений пароль.

Старий пароль порівнюється з паролем за замовчуванням і лише в цьому випадку дозволяється зміна пароля.

Якщо розглядати форму-зону адміністратора то потрібно постійно аналізувати натиснуті кнопки та змінювати видимість Visibility.Hiden Visibility.Visible для різних компонентів. При натисканні на ту чи іншу кнопку продовжити ми переходимо до основної частини, де записані поля перевіряються регулярними виразами. Приклад для частини коду де перевіряється коректність введення спільних полів для викладача та студента:

if (Menu1.SelectedIndex == 0 | Menu1.SelectedIndex == 1)//lecturer or student

{

ChangeButton.Visibility = Visibility.Hidden;

Regex reg = new Regex(@"[А-ЯЇІЄЙ]([а-яіїяюєй]){1,}");

Match m = reg.Match(SurnameText.Text); //Прізвище

if (m.Success)

{

resulting1.Foreground = Brushes.Green;

resulting1.Content = "OK";

helping++;

}

else

{

resulting1.Foreground = Brushes.Red;

resulting1.Content = "Неправильно";

}

reg = new Regex(@"[А-ЯЇІЄЙ]([а-яіїяюєй]){1,}");

m = reg.Match(NameText.Text); //Ім’я

if (m.Success)

{

resulting2.Foreground = Brushes.Green;

resulting2.Content = "OK";

helping++;

}

else

{

resulting2.Foreground = Brushes.Red;

resulting2.Content = "Неправильно";

}

reg = new Regex(@"[А-ЯЇІЄЙ]([а-яіїяюєй]){1,}");

m = reg.Match(FatherNameText.Text); //По батькові

if (m.Success)

{

resulting3.Foreground = Brushes.Green;

resulting3.Content = "OK";

helping++;

}

else

{

resulting3.Foreground = Brushes.Red;

resulting3.Content = "Неправильно";

}

reg = new Regex(@"((((0[1-9])|(1\d)|(2[01234]))\.(01|02|03|04|05|06|07|08|09|10|11|12))|(2[56789]|30)\.(01|03|04|05|06|07|08|09|10|11|12)|(31\.(01|03|05|07|08|10|12)))\.((19[1-9]\d)|(200[012]))");

m = reg.Match(BirthdayText.Text); //Дата народження

if (m.Success)

{

resulting4.Foreground = Brushes.Green;

resulting4.Content = "OK";

helping++;

}

else

{

resulting4.Foreground = Brushes.Red;

resulting4.Content = "Неправильно";

}

reg = new Regex(@"(196[2-9]|19[789]\d|200\d|201[0-7])");

m = reg.Match(YearStartText.Text); //Рік старт

if (m.Success)

{

resulting5.Foreground = Brushes.Green;

resulting5.Content = "OK";

helping++;

}

else

{

resulting5.Foreground = Brushes.Red;

resulting5.Content = "Неправильно";

}

reg = new Regex(@"\d{10}");

m = reg.Match(KodText.Text); //Ідентифікаціний код

if (m.Success)

{

resulting6.Foreground = Brushes.Green;

resulting6.Content = "OK";

helping++;

}

else

{

resulting6.Foreground = Brushes.Red;

resulting6.Content = "Неправильно";

}

reg = new Regex(@"(Ф((ІМ)|(ІКТ))|Ф((ЕМ)|(ОФ)))|(ГЕФ)");

m = reg.Match(FacultyText.Text); //Факультет

if (m.Success)

{

resulting7.Foreground = Brushes.Green;

resulting7.Content = "OK";

helping++;

}

else

{

resulting7.Foreground = Brushes.Red;

resulting7.Content = "Неправильно";

}

Якщо регулярний вираз записано коректно то в сусідньому Label стає «ОК», що стає сигналом для додавання одиниці для спеціальної контролюючої змінної helping, лише якщо всі поля мають зміст «ОК» виконається наступна дія.

Наприклад функції зміни запису спочатку перевіряється існування такого запису, та відбувається його видалення та додавання. Приклад розглянуто в такому коді:

uint KOD;

string path\_2 = @"course\_project1.txt";

if (uint.TryParse(KodText.Text, out KOD))

{

if (DeleteSTUDENTFromFile(KOD))

{ MessageBox.Show("Елемент типу СТУДЕНТ було успішно видалено з бази даних!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information); }

}

if (SurnameText.Text != null && NameText.Text != null && FatherNameText.Text != null && BirthdayText.Text != null &&YearStartText.Text != null && KodText.Text != null && FacultyText.Text != null && GroupNameText.Text != null &&OwnMarkText.Text != null && CourseText.Text != null && helping == 10)

{

double tmpmark;

int tmpcourse = -1;

if (uint.TryParse(KodText.Text, out KOD))

{

if (int.TryParse(CourseText.Text, out tmpcourse))

{

if (double.TryParse(OwnMarkText.Text, out tmpmark))

{

Add\_And\_Print\_PointSTUDENTToDictionary(path\_2, GroupNameText.Text, tmpcourse, tmpmark, NameText.Text, SurnameText.Text, FatherNameText.Text,

KOD, BirthdayText.Text, YearStartText.Text, FacultyText.Text);

MessageBox.Show("Елемент типу СТУДЕНТ було успішно змінено в базі даних!", "Повідомлення!", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

}

При записі після знищення іншого запису з тим самим кодом відбувається копіювання потрібних записів поступово в інший тимчасовий файл та файл, що є основним перезаписується змістом тимчасового. Пізніше тимчасовий файл буде використовуватись для виведення у нього інформації для звичайного користувача. Приклад:

string path\_1 = @"G:\KursovaFleshechna\WPFMain\bin\Debug\course\_project1.txt";

string path\_ = @"G:\KursovaFleshechna\WPFMain\bin\Debug\course\_project.txt";

if (File.Exists(path\_1))

{

StreamReader sr = new StreamReader(path\_1, System.Text.Encoding.GetEncoding(1251));

string[] lines = { sr.ReadToEnd() };

MessageBox.Show("Вдало відкрили, считали!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information);

if (File.Exists(path\_))

{

File.WriteAllLines(path\_, lines);

MessageBox.Show("Вдало відкрили, перезаписали!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

Для зчитування використано функцію StreamReader та StreamWriter зі стандартної бібліотеки Sysyem.IO.Прклад розглянуто у функції для додавання елементу типу студент:

public Dictionary<uint,Student> Add\_And\_Print\_PointSTUDENTToDictionary(string pyt, string group,int course, double averageMark, string name, string surName,string secondName, uint iD\_Kod, string dateBirthday, string yearStart, string facultyName)

{ Dictionary<uint,Student> dict = new Dictionary<uint,Student>();

Student student=new Student(name,surName,secondName,iD\_Kod,dateBirthday,

yearStart,facultyName,group,course,averageMark);

dict.Add(iD\_Kod, student);

using (StreamWriter sw = new StreamWriter(pyt, true, System.Text.Encoding.GetEncoding(1251)))

{ sw.WriteLine("СТУДЕНТ");

sw.WriteLine("Прізвище:");

sw.WriteLine(surName);

sw.WriteLine("Ім’я:");

sw.WriteLine(name);

sw.WriteLine("По-батькові:");//path\_

sw.WriteLine(secondName);

sw.WriteLine("Дата народження:");

sw.WriteLine(dateBirthday);

sw.WriteLine("Рік вступу:");

sw.WriteLine(yearStart);

sw.WriteLine("Ідентифікаційний код:");

sw.WriteLine(iD\_Kod);

sw.WriteLine("Факультет:");

sw.WriteLine(facultyName);

sw.WriteLine("Курс:");

sw.WriteLine(course);

sw.WriteLine("Назва групи:");

sw.WriteLine(group);

sw.WriteLine("Середній бал:");

sw.WriteLine(averageMark);

} return dict;

}

В функції вказується шлях до файлу в який будуть записувати, потім вказують чи будуть дописувати чи переписувати, потім ми вказуємо кодування для читання чи запису певної строки у файл.

В ході другого розділу було спроектовано загальний алгоритм роботи програми та допоміжні алгоритми, в тому числі для роботи з файлами. Показали створення класів та вікон. Також проаналізували найважливіші ділянки коду.

Було спроектовано бібліотеку класів та використано максимально можливості однієї функції для іншої.

# РОЗДІЛ 3 ОПИС РОБОТИ З ПРОГРАМНИМ ДОДАТКОМ ТА ЙОГО ТЕСТУВАННЯ

## 3.1 Опис роботи додатку

На даному етапі курсової роботи буде продемонстрована робота вже реалізованої БД, а також протестовано можливості при стандартній ситуації та при можливості некоректного введення даних.

Тож розпочнемо тестування ілюструючи його відповідними рисунками.

При вході в БД користувач обирає кнопку залежно від потрібних функцій та можливостей доступу. Цю ситуацію зображено на рисунку 3.1.

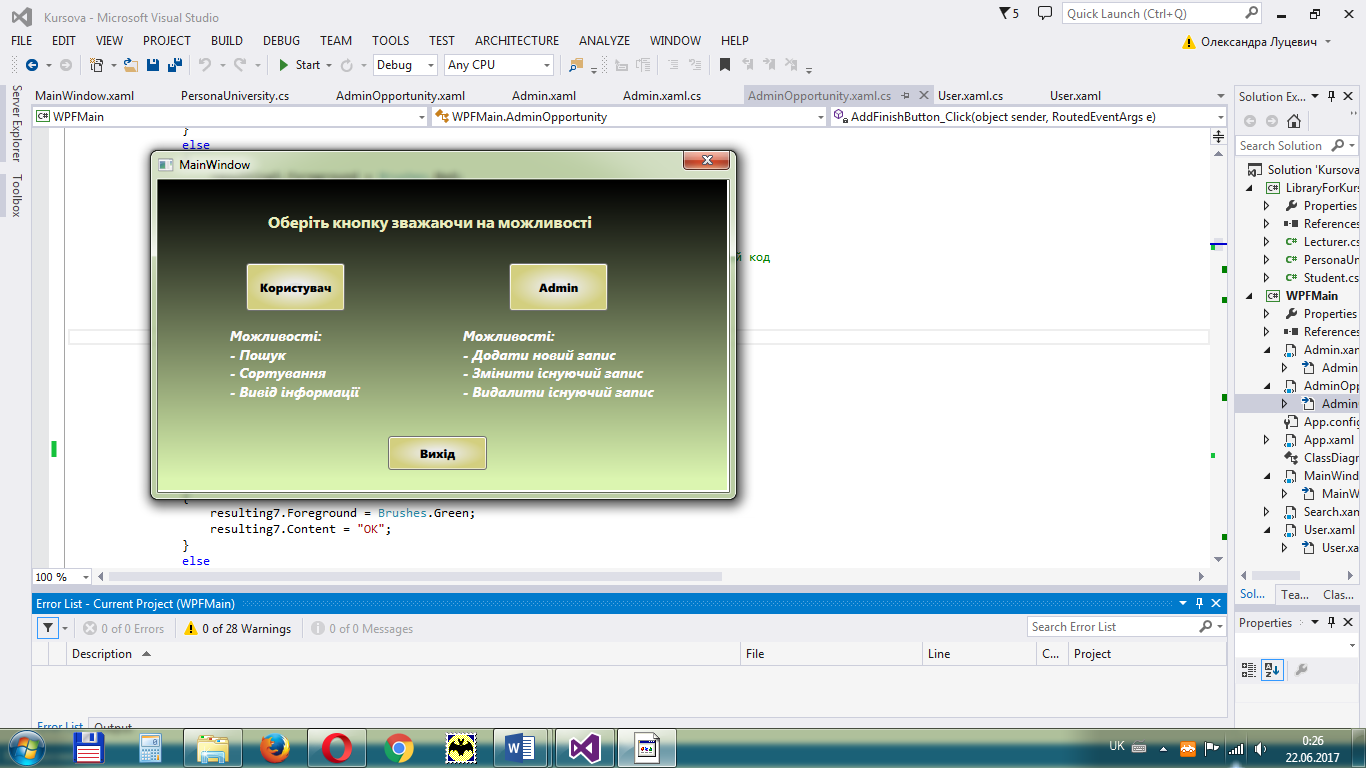


Рисунок 3.1 – Головне меню

При вході в зону адміністратора він повинен ввести коректний пароль, можлива функція підказки, кількість введень паролю в часі не обмежується. Ситуація зображена на рисунку 3.2.

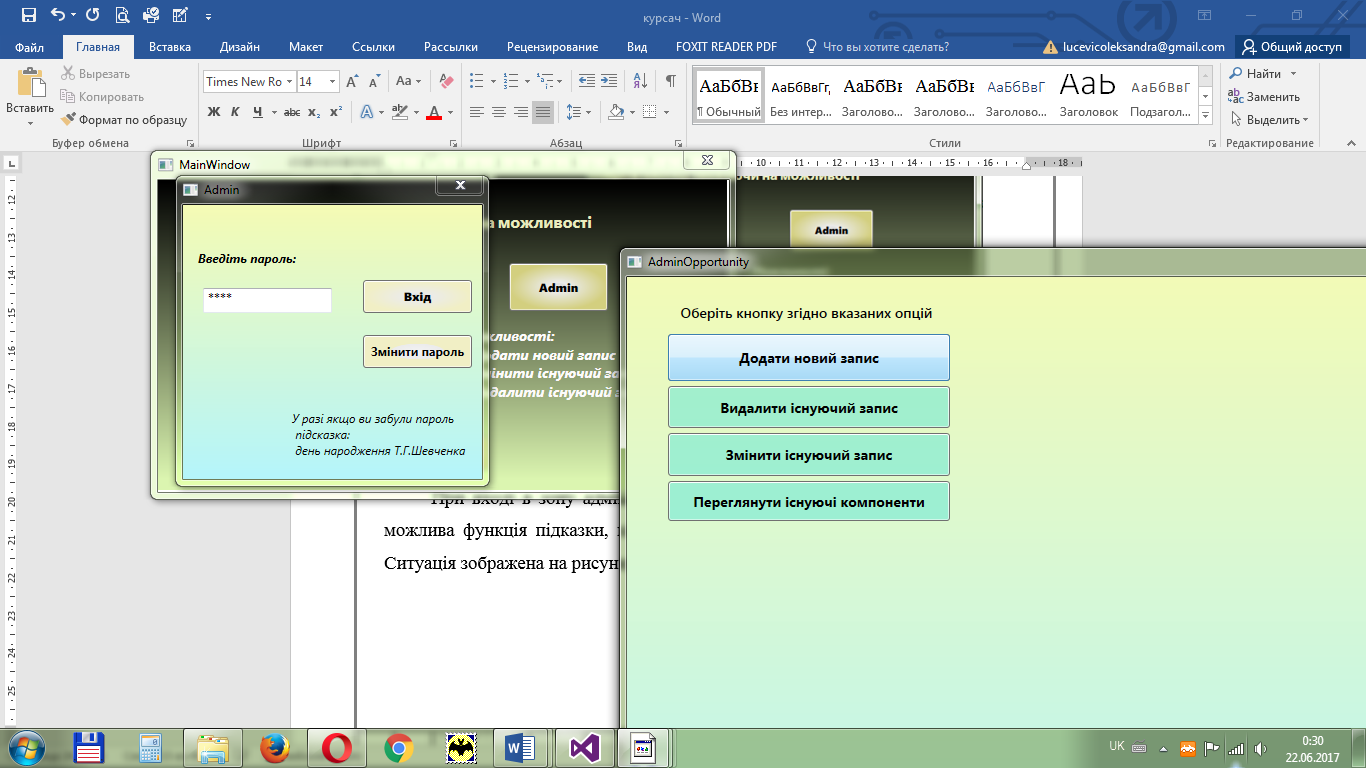


Рисунок 3.2 – Введення паролю та підказка

Після цього адміністратор може використати будь-яку кнопку яка показана на рисунку 3.3.

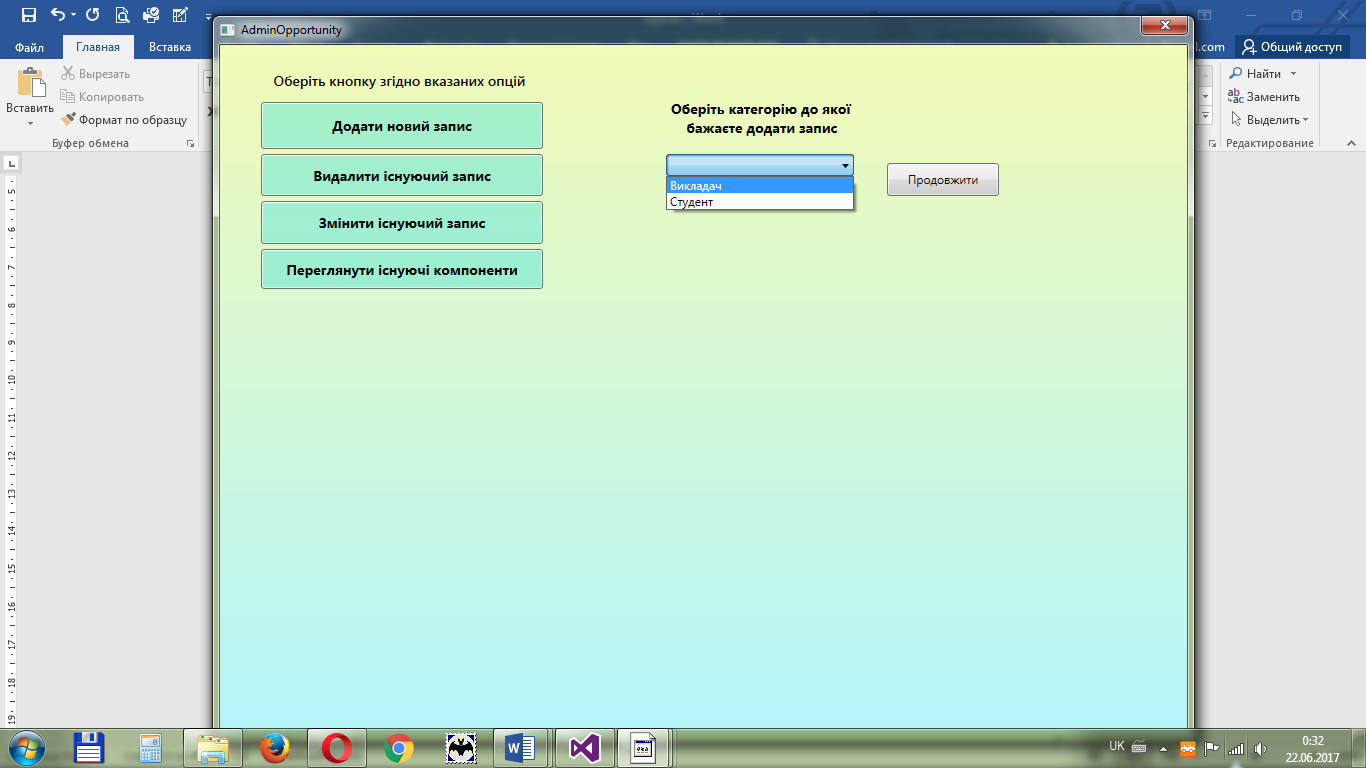


Рисунок 3.3 – Вибір додавання запису та попередній запит для розмежування зони Студент чи Викладач

Далі дані перевіряються регулярними виразами та надається коректний приклад як показано на рисунку 3.4.

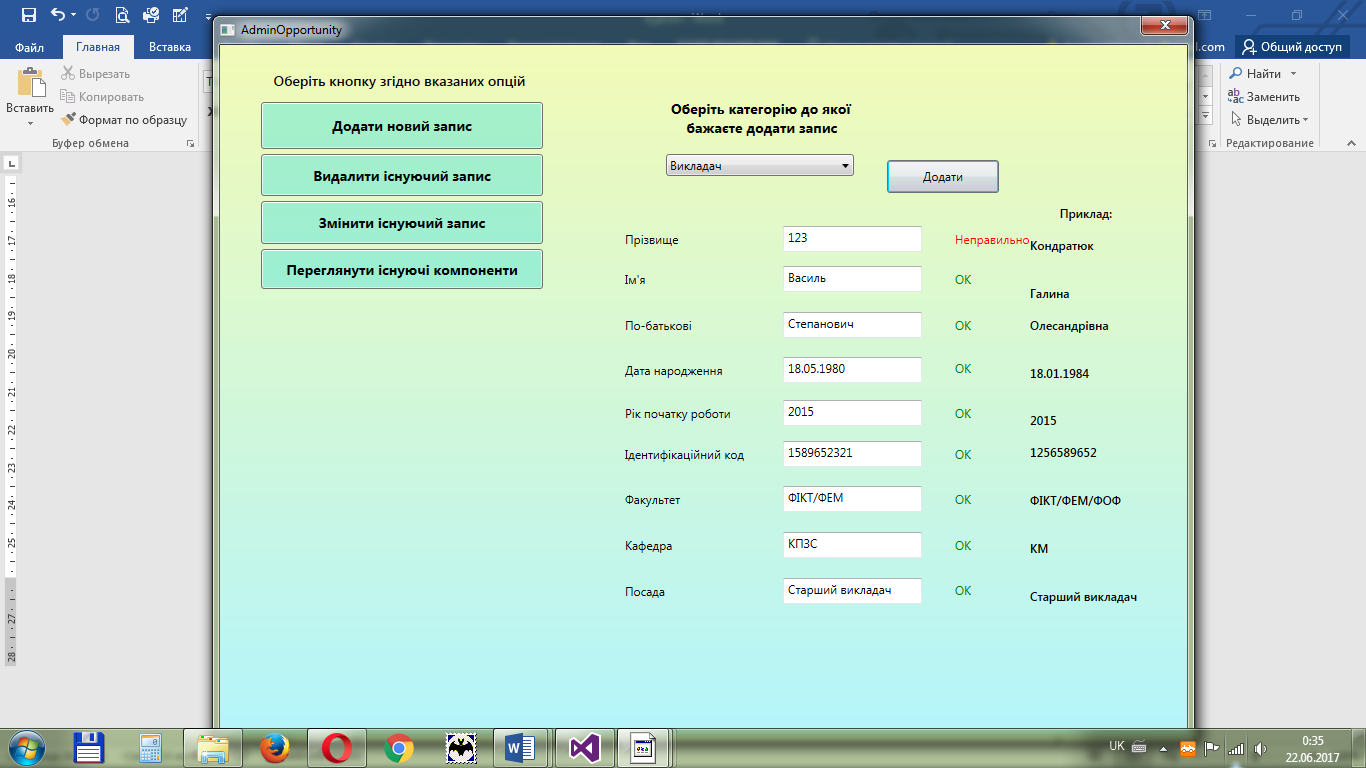


Рисунок 3.4 – Введення та контроль даних для додавання запису

Потім з’являється повідомлення про успішне виконання як показано на рисунку 3.5.

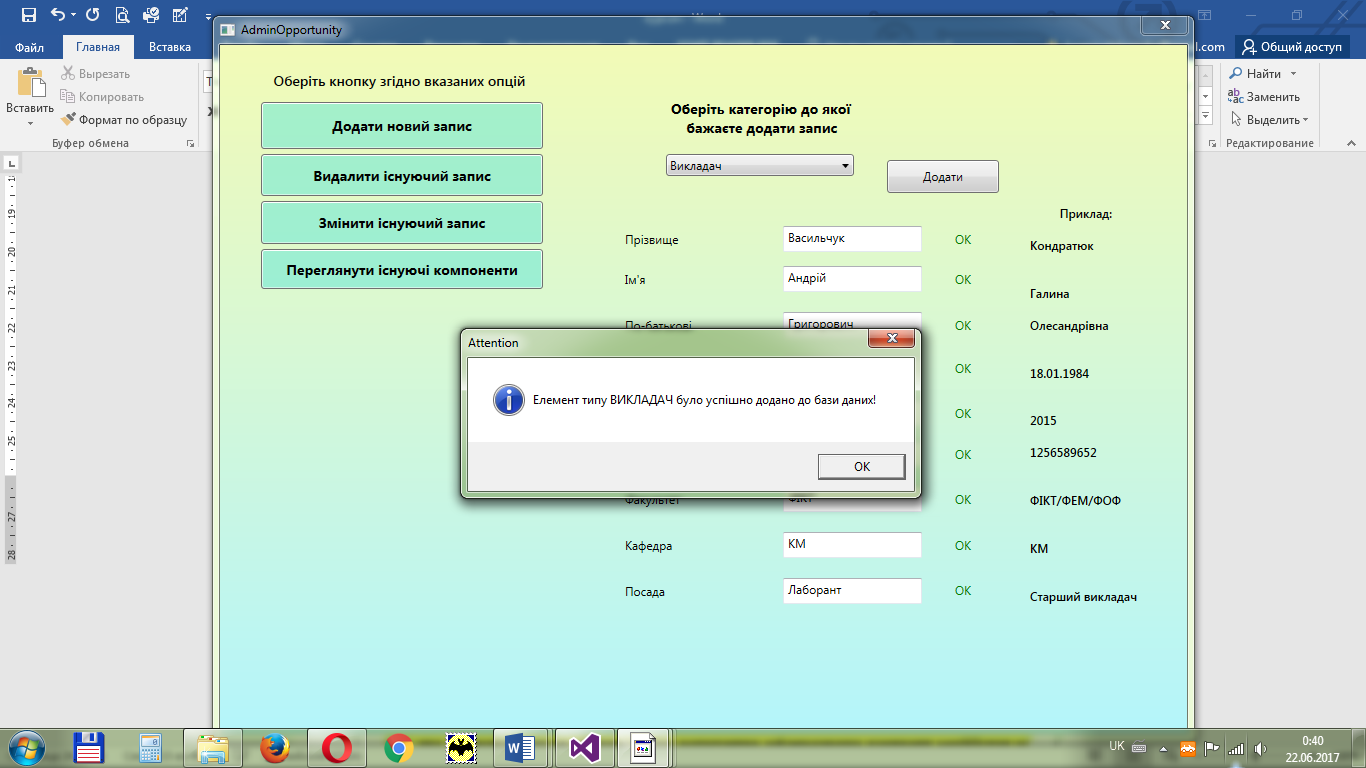
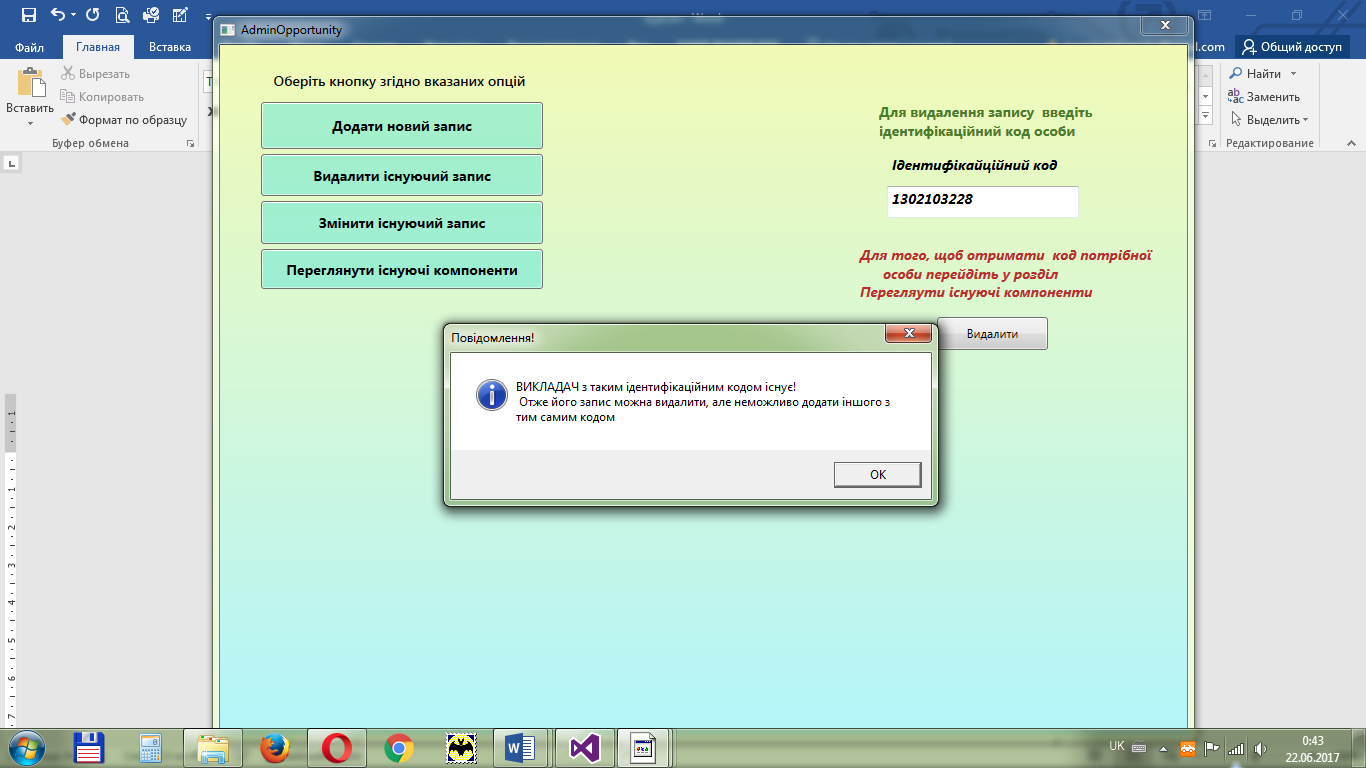


Рисунок 3.5 – Введення та контроль даних для додавання запису

Для видалення запису потрібно ввести ідентифікаційний код особи яку бажаємо видалити, це змушує адміністратора попередньо підготуватись та скопіювати код з файлу. Показано на рисунку 3.6.



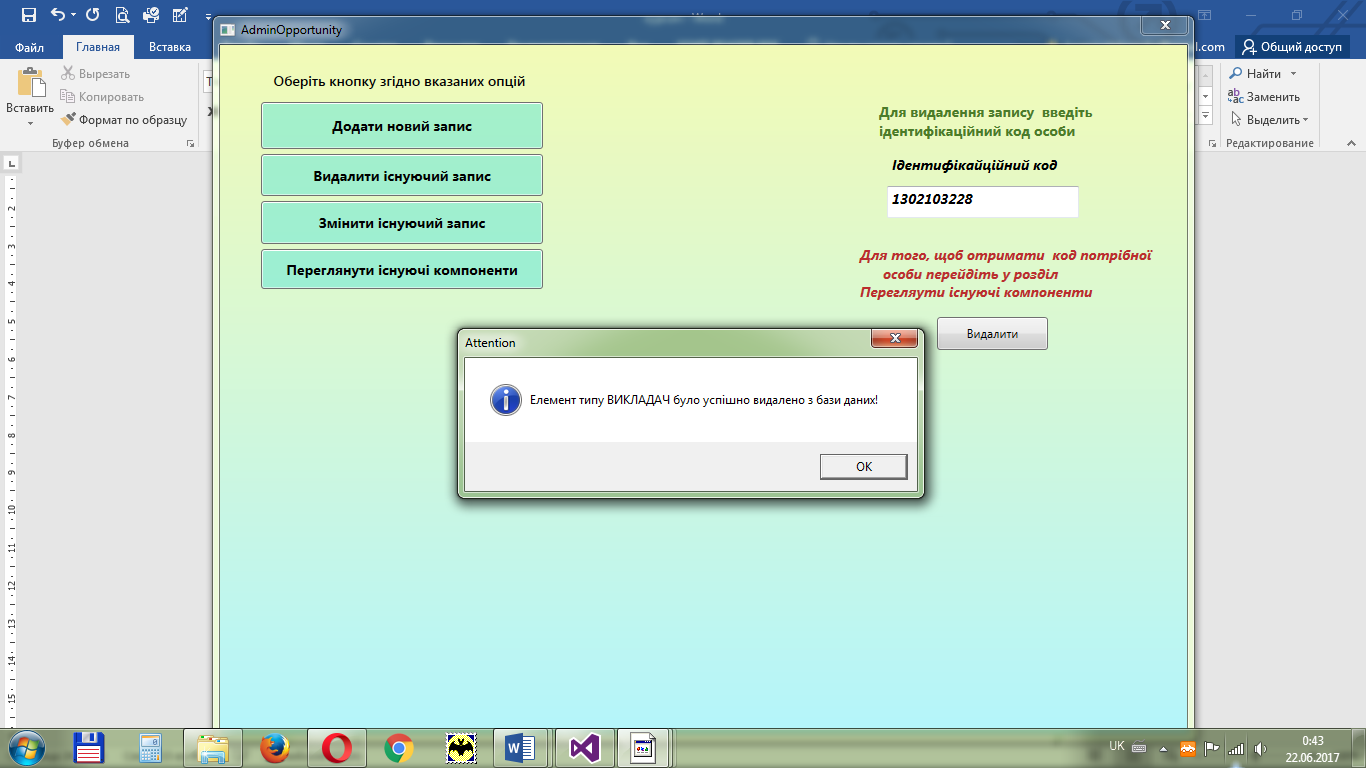


Рисунок 3.6 – Виведення відповідних повідомлень

Для зміни запису відбуваються аналогічні дії до додавання (перевірка на попереднє існування, видалення, створення нового запису) і тому не має потреби в ілюструванні цієї частини.

При натисканні на кнопку переглянути доступні компоненти ми отримуємо інформацію у відповідному файлі як показано на рисунку 3.7.

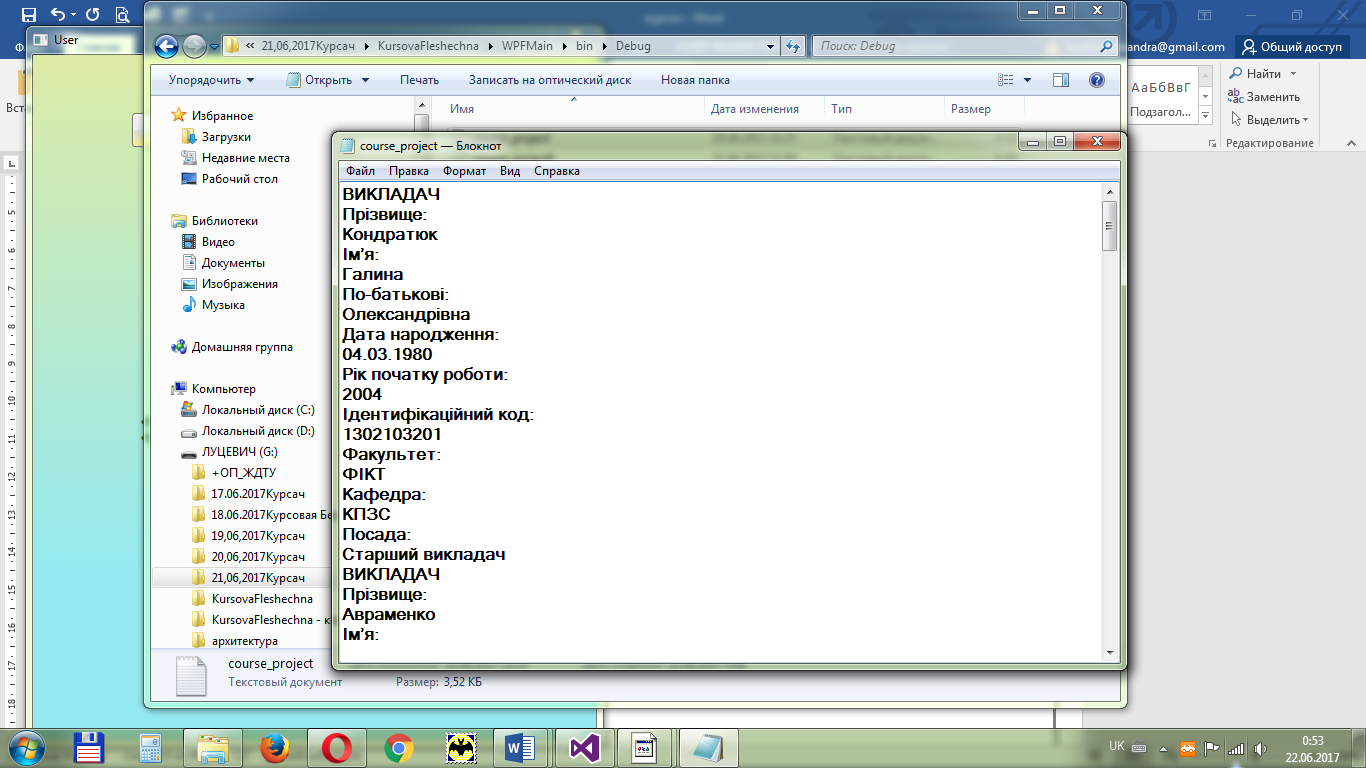


Рисунок 3.7 – Перегляд існуючої інформації у файлі

Якщо ж користувач натисне кнопку «Користувач» отримаємо доступ до даних, можливість пошуку та сортування даних. Такі ситуації розглянуто на рисунках 3.8-3.11.

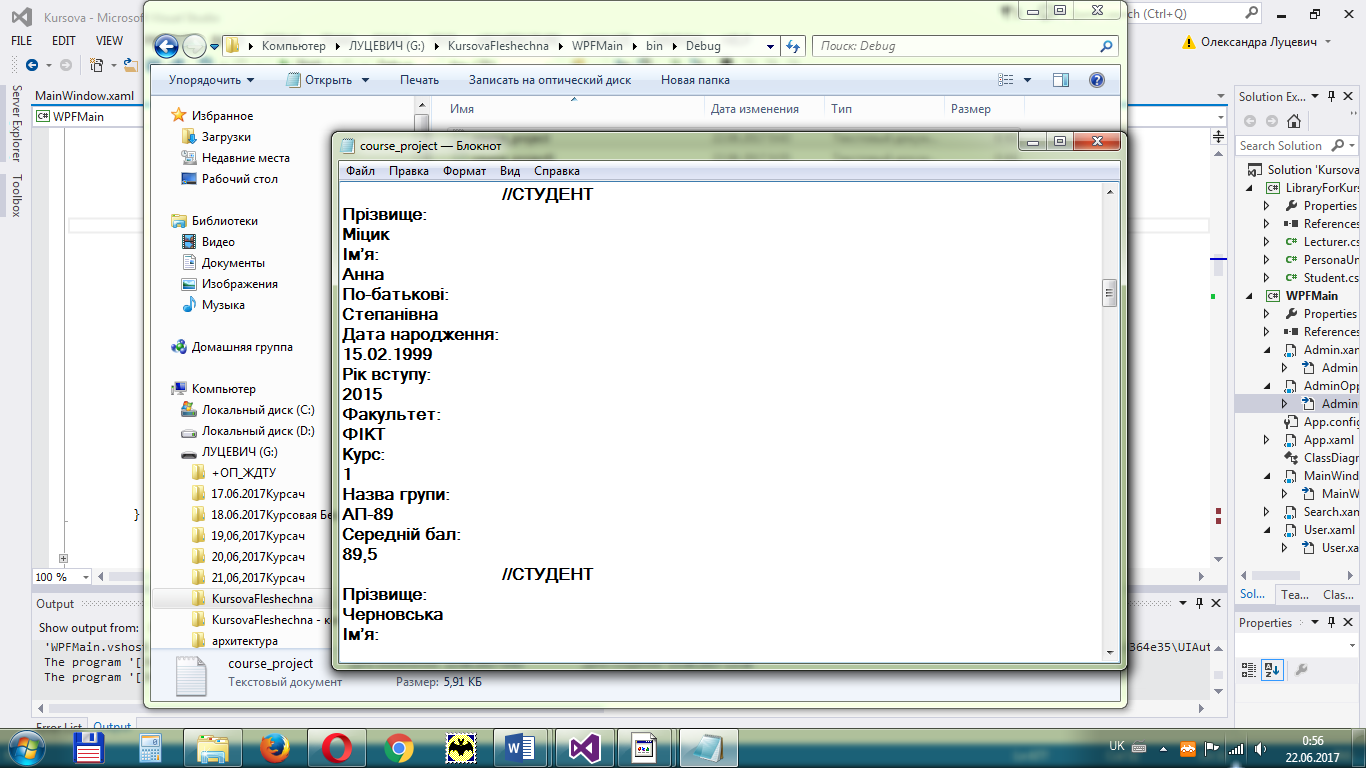


Рисунок 3.8 – Сортування за критерієм тільки студенти у файлі

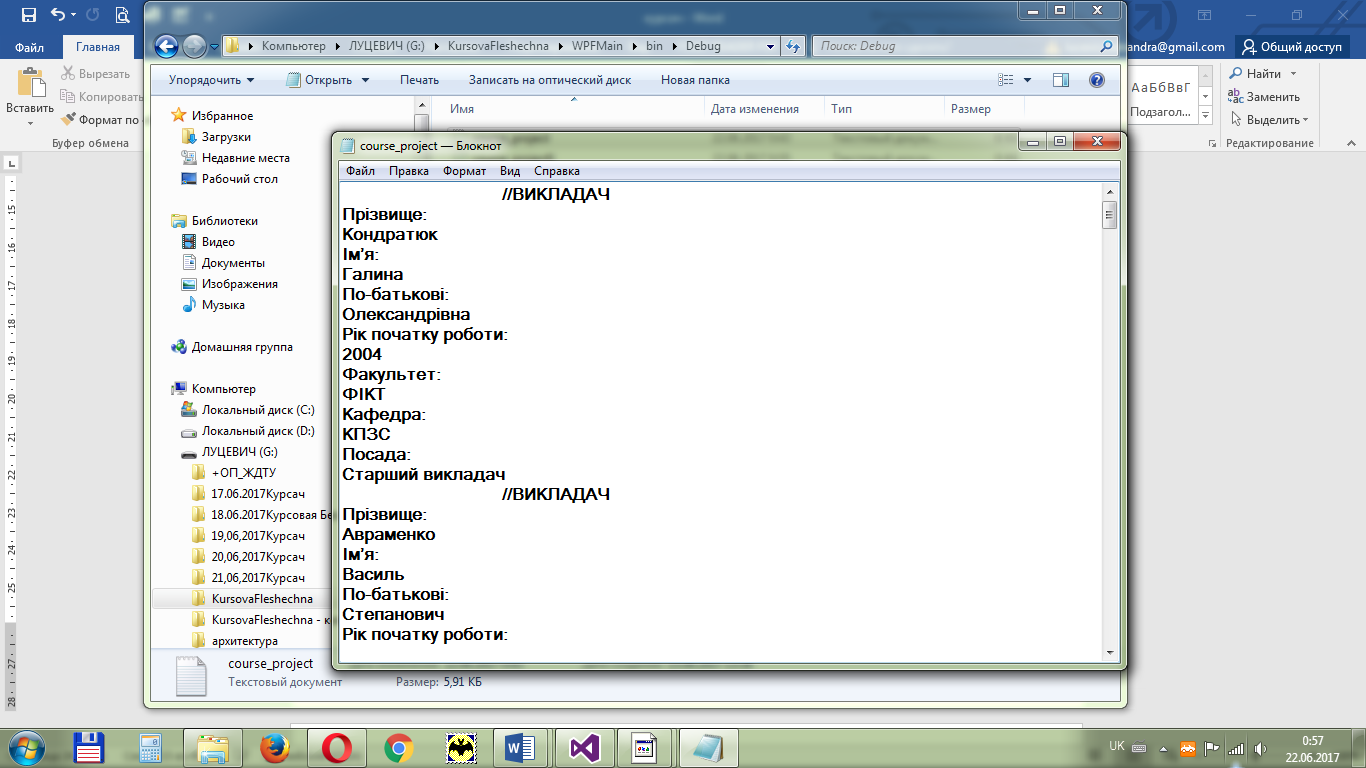


Рисунок 3.9 – Сортування за критерієм тільки викладачі у файлі

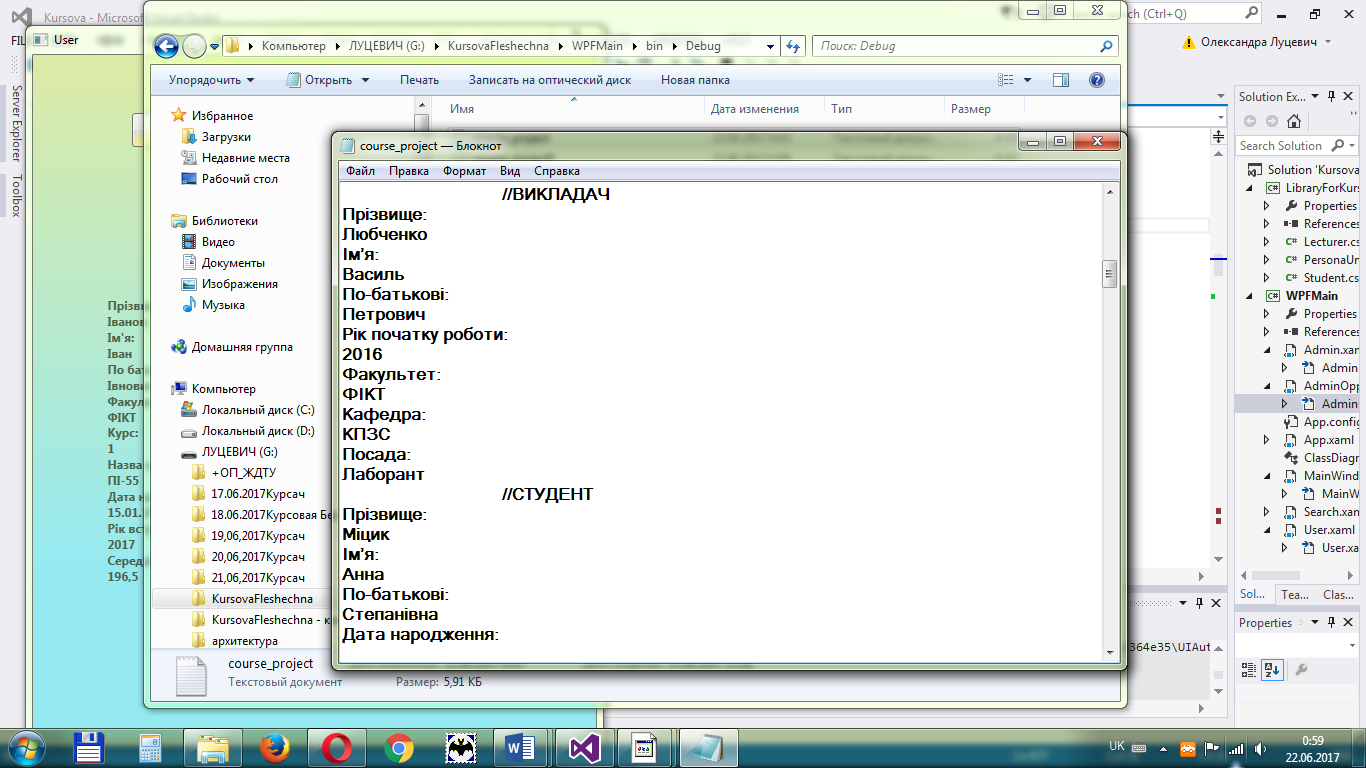


Рисунок 3.10 – Перегляд всіх записів у файлі

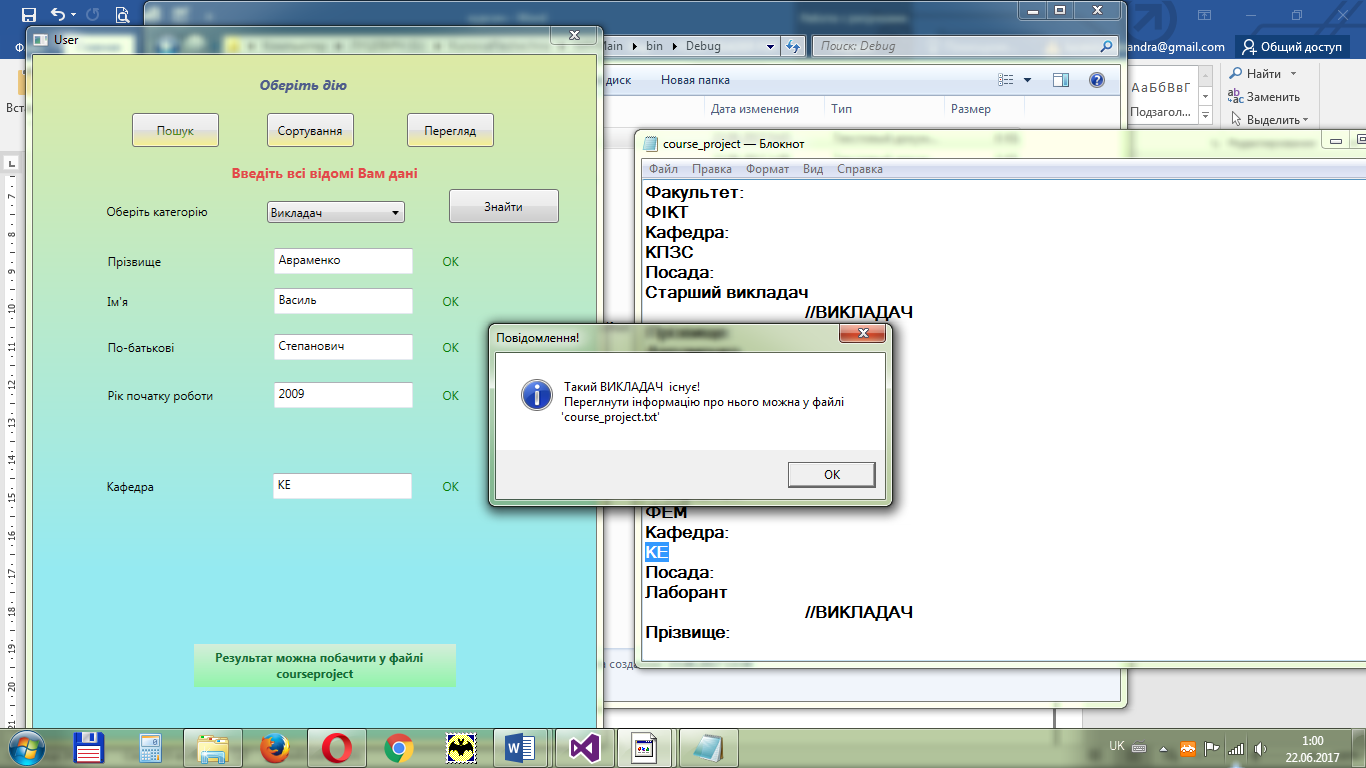


Рисунок 3.11 – Пошук і подальше читання інформації про шукану особу

В цьому розділі було протестовано БД «структура університету» та зображено часті помилки для коритсувача та обхід цих помилок та можливість опрацьовування цих помилок.

# ВИСНОВКИ

В першому розділі курсового проекту, було проаналізовано знайдені БД зі схожим функціоналом та інтерфейсом, та було сформовано уявлення про БД, що розроблюється. Було виявлено основні потреби для написання цієї БД.

В другому розділі курсового проекту, було проведено розробку функціональних і загальних частин БД. При розробці функціонального алгоритму розробили уявлення про методи розробки програмного коду додатку. На заключному етапі було розроблено програмний код спроектованого додатку. В результаті було отримано готовий програмний продукт який виконував всі заплановані функції для усіх можливостей.

В третьому розділі курсового проекту було описана методи користування створеної БД, були пояснені і продемонстровані всі створені елементи інтерфейсу в формі. Також було проведено тестування, тобто БД було запущено в реальних умовах її користування. В ході тестування проблем не виникало.

В результаті виконання курсового проекту проекту отримано БД, яка повністю відповідає запланованому функціоналу поставленим на початку проектування вимогам. БД є простою та зручною.

Спроектоване БД в повній мірі відповідає поставленим вимогам та є достатньо швидкодіючим.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Dictionary<TKey, TValue>.ContainsKey Method (TKey)- 2016. [Електронний ресурс] – Режим доступу:

<https://msdn.microsoft.com/en-us/library/kw5aaea4(v=vs.110).aspx>

2) Елементи управління WPF– 2015. [Електронний ресурс] – Режим доступу: <https://metanit.com/sharp/wpf.php>

3) Практическое руководство. Запись текста в файл – 2015. [Електронний ресурс] – Режим доступу:

<https://msdn.microsoft.com/ru-u/library/6ka1wd3w(v=vs.110).aspx>

4) Метод File.Copy– 2016. [Електронний ресурс] – Режим доступу: <https://msdn.microsoft.com/ru-ru/library/9706cfs5(v=vs.110).aspx>

5) Перемещение, копирование и удаление файлов– 2017. [Електронний ресурс] – Режим доступу:

<https://professorweb.ru/my/csharp/thread_and_files/level3/3_6.php>

# ДОДАТКИ

Додаток А

Програмний код

PersonaUniversity.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LibraryForKursovaya

{

public abstract class PersonaUniversity

{

public string Name;//

public string SurName;//

public string SecondName;//

public uint ID\_Kod;//

public string FacultyName;

public string DateBirthday;//

public string YearStart;

public void SetName(string name)

{

if (name.Length > 1)

Name = name;

}

public string GetName()

{

return Name;

}

public void SetSurName(string surname)

{

if (surname.Length > 1)

SurName = surname;

}

public string GetSurName()

{

return SurName;

}

public void SetSecondName(string secondname)

{

if (secondname.Length > 4)

SecondName = secondname;

}

public string GetSecondName()

{

return SecondName;

}

public void SetID\_Kod(uint id\_kod)

{

if (id\_kod > 1000000000)

ID\_Kod = id\_kod;

}

public uint GetID\_Kod()

{

return ID\_Kod;

}

public void SetFaculty(string facultyName)

{

if (facultyName.Length >= 3)

FacultyName = facultyName;

}

public string GetFaculty()

{ return FacultyName; }

public void SetDateBirthday(string dateBirthday)

{

if (dateBirthday.Length ==10)

DateBirthday = dateBirthday;

}

public string GetDateBirthday()

{

return DateBirthday;

}

public void SetYearStart(string yearStart)

{

if (yearStart.Length ==4)

YearStart = yearStart;

}

public string GetYearStart()

{

return YearStart;

}

public PersonaUniversity(string name, string surName, string secondName, uint id\_kod, string dateBirthday, string year, string facultyName)

{

SetName(name);

SetSurName(surName);

SetSecondName(secondName);

SetID\_Kod(id\_kod);

SetDateBirthday(dateBirthday);

SetYearStart(year);

SetFaculty(facultyName);

}

}

}

Lecturer.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LibraryForKursovaya

{

public class Lecturer: PersonaUniversity

{

public string Chair;

public string Position;

public Lecturer(string name, string surName, string secondName, uint id\_kod, string dateBirthday, string year, string facultyName, string chair, string position)

: base(name,surName,secondName,id\_kod,dateBirthday,year,facultyName)

{

SetChair(chair);

SetPosition(position);

}

public void SetChair(string chair)

{

if (chair.Length > 2 && chair.Length < 6)

Chair = chair;

}

public string GetChair()

{

return Chair;

}

public void SetPosition(string position)

{

if (position.Length > 0 && position.Length < 15)

Position = position;

}

public string GetPosition()

{

return Position;

}

}

}

Student.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LibraryForKursovaya

{

public class Student:PersonaUniversity

{

public string Group;

public int Course;

public double AverageMark;

public Student (string name, string surName, string secondName, uint id\_kod, string dateBirthday, string year, string facultyName,string group, int course, double averageMark)

: base(name,surName,secondName,id\_kod,dateBirthday,year,facultyName)

{

SetGroup(group);

SetCourse(course);

SetAverageMark(averageMark);

}

public void SetGroup(string group)

{

if (group.Length > 4 && group.Length <=6)

Group = group;

}

public string GetGroup()

{

return Group;

}

public void SetCourse(int course)

{

if (course>0 && course<6)

Course = course;

}

public int GetCourse()

{

return Course;

}

public void SetAverageMark(double averageMark)

{

if (averageMark > 0 && averageMark <=100 )

AverageMark = averageMark;

}

public double GetAverageMark()

{

return AverageMark;

}

}

}

Admin.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace WPFMain

{

/// <summary>

/// Interaction logic for Admin.xaml

/// </summary>

public partial class Admin : Window

{

public string Password;

public bool Oznaka=false;

public Admin()

{

InitializeComponent();

Label1.Visibility = Visibility.Hidden;

Help.Visibility = Visibility.Hidden;

Password\_Copy1.Visibility = Visibility.Hidden;

Password\_Copy2.Visibility = Visibility.Hidden;

PasswordTB\_Copy.Visibility = Visibility.Hidden;

PasswordTB\_Copy1.Visibility = Visibility.Hidden;

PasswordTB.MaxLength = 8;

PasswordTB.HorizontalContentAlignment = HorizontalContentAlignment;

}

private void PasswordButton\_Click(object sender, RoutedEventArgs e)

{

Password\_Copy1.Visibility = Visibility.Visible;

Password\_Copy2.Visibility = Visibility.Visible;

Password = "1814";

Help.Visibility = Visibility.Visible;

PasswordTB\_Copy.Visibility = Visibility.Visible;

PasswordTB\_Copy1.Visibility = Visibility.Visible;

string PasswordOne = PasswordTB\_Copy.Password;

string PasswordTwo = PasswordTB\_Copy1.Password;

if (PasswordOne == Password)

{

Password = PasswordTwo;

Label1.Content = "Ви змінили пароль!";

Oznaka = true;

Label1.Visibility = Visibility.Visible;

Help.Visibility = Visibility.Hidden;

AdminOpportunity frm = new AdminOpportunity();

frm.Show();

}

}

private void Enter\_Click(object sender, RoutedEventArgs e)

{

if (!Oznaka)

{ Password = "1814"; }

string copy = PasswordTB.Password;

if (copy== Password)

{

AdminOpportunity frm = new AdminOpportunity();

frm.Show();

}

else

if(!Oznaka)

Label1.Visibility = Visibility.Visible;

}

}

}

User.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.IO;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.Text.RegularExpressions;

using System.Web.RegularExpressions;

using LibraryForKursovaya;

namespace WPFMain

{

/// <summary>

/// Interaction logic for User.xaml

/// </summary>

public partial class User : Window

{

public string path\_ = @"course\_project.txt";

public string path\_1 = @"course\_project1.txt";

public User()

{

InitializeComponent();

HiddenAll();

Menu.Visibility = Visibility.Hidden;

Found1.Visibility = Visibility.Hidden;////////

Found2.Visibility = Visibility.Hidden;////////

FoundButton1.Visibility = Visibility.Hidden;

FoundButton2.Visibility = Visibility.Hidden;

}

public bool ReadSTUDENT\_AndFoundFromFile( string NAME,string SURNAME,

string SECONDNAME,string COURSE,string FACULTY, string GROUP,string YEARSTART)

{

using (StreamReader sr = new StreamReader(path\_, System.Text.Encoding.Default))

{

string[] lines = { };

if (File.Exists(path\_1))

{

File.WriteAllLines(path\_1, lines);

}

int amount = 0, amount1 = 0;

if (NAME != null) {amount++;}

if (SURNAME != null) {amount++;}

if (SECONDNAME != null) {amount++;}

if (FACULTY != null) {amount++;}

if (YEARSTART != null) {amount++;}

if (GROUP != null) {amount++;}

if (COURSE != null) { amount++; }

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", groupname = "//";

int course = -1;

double averagemark = 1.0;

string facultyName = "";

uint kod = 1000000000;

bool oznaka1 = false;

line = sr.ReadLine();

if (line == null)

{

return false;

}

do

{

if (line == null)

{

return true;

}

if (line == "СТУДЕНТ" && ((line = sr.ReadLine()) != null))

{

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік вступу:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Курс:")) //

{ course = Convert.ToUInt16(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Назва групи:")) //

{ groupname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Середній бал:")) //

{ averagemark = Convert.ToDouble(sr.ReadLine()); }

if (NAME==name){amount1++;}

if (SURNAME==surname){amount1++;}

if (SECONDNAME==secondname){amount1++;}

if (FACULTY==facultyName){amount1++;}

if (GROUP==groupname){amount1++;}

if (YEARSTART==yearstart){amount1++;}

if (COURSE == course.ToString()) { amount1++; }

if(amount1!=0&&amount1<=amount)

{

WRITELINE(path\_1, "\t\t//СТУДЕНТ");

WRITELINE(path\_1, "Прізвище:");

WRITELINE(path\_1, surname);

WRITELINE(path\_1, "Ім’я:");

WRITELINE(path\_1, name);

WRITELINE(path\_1, "По-батькові:");

WRITELINE(path\_1, secondname);

WRITELINE(path\_1, "Дата народження:");

WRITELINE(path\_1, datebirthday);

WRITELINE(path\_1, "Рік вступу:");

WRITELINE(path\_1, yearstart);

WRITELINE(path\_1, "Факультет:");

WRITELINE(path\_1, facultyName);

WRITELINE(path\_1, "Курс:");

WRITELINE(path\_1, course.ToString());

WRITELINE(path\_1, "Назва групи:");

WRITELINE(path\_1, groupname);

WRITELINE(path\_1, "Середній бал:");

WRITELINE(path\_1, averagemark.ToString());

//

}

}

else

{

if (line == "ВИКЛАДАЧ")

{

for (int I = 1; I <= 18; I++)

{ line = sr.ReadLine(); }

}

}

line = sr.ReadLine();

} while (!oznaka1);

return true;

}

}

public bool ReadLECTURERS\_AndFoundFromFile(string NAME, string SURNAME,

string SECONDNAME, string YEARSTART, string CHAIR)//&&&&&&&&&

{

using (StreamReader sr = new StreamReader(path\_, System.Text.Encoding.GetEncoding(1251)))

{

int amount = 0, amount1 = 0;

string[] lines = { };

if (NAME != null) { amount++; }

if (SURNAME != null) { amount++; }

if (SECONDNAME != null) { amount++; }

if (CHAIR != null) { amount++; }

if (YEARSTART != null) { amount++; }

if (File.Exists(path\_1))

{

File.WriteAllLines(path\_1, lines);

}

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", chair = "//", position = "//";

string facultyName = "";

uint kod = 1000000000;

bool oznaka1 = false;

line = sr.ReadLine();

if (line == null)

{

return false;//если сразу пусто то фолс

}

do

{

if (line == null)//если дошел до конца файла то верни количество ПОСЛЕ

{

oznaka1 = true;

return true;

}

if (line == "СТУДЕНТ")//если это преподователь то считай его поля

{

for (int I = 1; I <= 20; I++)

{ line = sr.ReadLine(); }

}

if (line == "ВИКЛАДАЧ")//если это преподователь то считай его поля

{

line = sr.ReadLine();

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік початку роботи:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Кафедра:")) //

{ chair = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Посада:")) //

{ position = sr.ReadLine(); }

if (NAME == name) { amount1++; }

if (SURNAME == surname) { amount1++; }

if (SECONDNAME == secondname) { amount1++; }

if (CHAIR == chair) { amount1++; }

if (YEARSTART == yearstart) { amount1++; }

if (amount1 != 0 && amount1 <= amount)

{

WRITELINE(path\_1, "\t\t//ВИКЛАДАЧ");

WRITELINE(path\_1, "Прізвище:");

WRITELINE(path\_1, surname);

WRITELINE(path\_1, "Ім’я:");

WRITELINE(path\_1, name);

WRITELINE(path\_1, "По-батькові:");

WRITELINE(path\_1, secondname);

WRITELINE(path\_1, "Рік початку роботи:");

WRITELINE(path\_1, yearstart);

WRITELINE(path\_1, "Факультет:");

WRITELINE(path\_1, facultyName);

WRITELINE(path\_1, "Кафедра:");

WRITELINE(path\_1, chair);

WRITELINE(path\_1, "Посада:");

WRITELINE(path\_1, position);

File.Copy(@"G:\KursovaFleshechna\WPFMain\bin\Debug\course\_project1.txt", @"G:\KursovaFleshechna\course\_project.txt", true);

}

}

line = sr.ReadLine();

} while (!oznaka1);

return true;

}

}

public Dictionary<uint, Lecturer> ReadLECTURER\_AndFoundFromFile(uint kod1)

{

using (StreamReader sr = new StreamReader(path\_, System.Text.Encoding.Default))

{

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", chair = "//", position = "//";

string facultyName = "ФІКТ";

uint kod = 1000000000;

bool oznaka1 = false;

Dictionary<uint, Lecturer> dict2 = new Dictionary<uint, Lecturer>();

line = sr.ReadLine();

if (line == null)

{

Lecturer lecturer2 = new Lecturer(name, surname, secondname, kod, datebirthday, yearstart, facultyName, chair, position);

dict2.Add(1000000000, lecturer2);

return dict2;

}

do

{

if (line == null)

{

Lecturer lecturer2 = new Lecturer(name, surname, secondname, kod, datebirthday, yearstart, facultyName, chair, position);

dict2.Add(1000000000, lecturer2);

oznaka1 = true;

}

if (line == "ВИКЛАДАЧ" && (line = sr.ReadLine()) != null)

{

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік початку роботи:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Кафедра:")) //

{ chair = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Посада:")) //

{ position = sr.ReadLine(); }

if (kod1 == kod)

{

Dictionary<uint, Lecturer> dict = new Dictionary<uint, Lecturer>();

Lecturer lecturer1 = new Lecturer(name, surname, secondname, kod, datebirthday, yearstart, facultyName, chair, position);

dict.Add(kod, lecturer1);

return dict;

}

}

else

{

if (line == "СТУДЕНТ")

{

for (int I = 1; I <= 20; I++)

{ line = sr.ReadLine(); }

}

}

line = sr.ReadLine();

} while (!oznaka1);

return dict2;

}

}

public void WRITELINE(string pyt, string word)

{

using (StreamWriter sw = new StreamWriter(pyt, true, System.Text.Encoding.GetEncoding(1251)))

{

sw.WriteLine(word);

}

}

public bool ReadSTUDENT\_LECTURERS\_and\_Write\_FromFile()//&&&&&&&&&

{

using ( StreamReader sr = new StreamReader(path\_, System.Text.Encoding.GetEncoding(1251)))

{

string[] lines = { };

if (File.Exists(path\_1))

{

File.WriteAllLines(path\_1, lines);

}

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", groupname = "//", chair = "//", position = "//";

int course = -1;

double averagemark = 1.0;

string facultyName = "";

uint kod = 1000000000;

bool oznaka1 = false;

line = sr.ReadLine();

if (line == null)

{

return false;//если сразу пусто то фолс

}

do

{

if (line == null)//если дошел до конца файла то верни количество ПОСЛЕ

{

oznaka1 = true;

return true;

}

if (line == "СТУДЕНТ")//если это преподователь то считай его поля

{

line = sr.ReadLine();

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік вступу:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Курс:")) //

{ course = Convert.ToUInt16(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Назва групи:")) //

{ groupname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Середній бал:")) //

{ averagemark = Convert.ToDouble(sr.ReadLine()); }

WRITELINE(path\_1, "\t\t//СТУДЕНТ");

WRITELINE(path\_1, "Прізвище:");

WRITELINE(path\_1, surname);

WRITELINE(path\_1, "Ім’я:");

WRITELINE(path\_1, name);

WRITELINE(path\_1, "По-батькові:");

WRITELINE(path\_1, secondname);

WRITELINE(path\_1, "Дата народження:");

WRITELINE(path\_1, datebirthday);

WRITELINE(path\_1, "Рік вступу:");

WRITELINE(path\_1, yearstart);

WRITELINE(path\_1, "Факультет:");

WRITELINE(path\_1, facultyName);

WRITELINE(path\_1, "Курс:");

WRITELINE(path\_1, course.ToString());

WRITELINE(path\_1, "Назва групи:");

WRITELINE(path\_1, groupname);

WRITELINE(path\_1, "Середній бал:");

WRITELINE(path\_1, averagemark.ToString());

}

if (line == "ВИКЛАДАЧ")//если это преподователь то считай его поля

{

line = sr.ReadLine();

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік початку роботи:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Кафедра:")) //

{ chair = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Посада:")) //

{ position = sr.ReadLine(); }

WRITELINE(path\_1, "\t\t//ВИКЛАДАЧ");

WRITELINE(path\_1, "Прізвище:");

WRITELINE(path\_1, surname);

WRITELINE(path\_1, "Ім’я:");

WRITELINE(path\_1, name);

WRITELINE(path\_1, "По-батькові:");

WRITELINE(path\_1, secondname);

WRITELINE(path\_1, "Рік початку роботи:");

WRITELINE(path\_1, yearstart);

WRITELINE(path\_1, "Факультет:");

WRITELINE(path\_1, facultyName);

WRITELINE(path\_1, "Кафедра:");

WRITELINE(path\_1, chair);

WRITELINE(path\_1, "Посада:");

WRITELINE(path\_1, position);

}

line = sr.ReadLine();

}while(!oznaka1);

return true;

}

}

public bool ReadSTUDENT\_and\_Write\_FromFile()//&&&&&&&&&

{

using (StreamReader sr = new StreamReader(path\_, System.Text.Encoding.GetEncoding(1251)))

{

string[] lines = { };

if (File.Exists(path\_1))

{

File.WriteAllLines(path\_1, lines);

}

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", groupname = "//";

int course = -1;

double averagemark = 1.0;

string facultyName = "";

uint kod = 1000000000;

bool oznaka1 = false;

line = sr.ReadLine();

if (line == null)

{

return false;//если сразу пусто то фолс

}

do

{

if (line == null)//если дошел до конца файла то верни количество ПОСЛЕ

{

oznaka1 = true;

return true;

}

if (line == "СТУДЕНТ")//если это преподователь то считай его поля

{

line = sr.ReadLine();

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік вступу:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Курс:")) //

{ course = Convert.ToUInt16(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Назва групи:")) //

{ groupname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Середній бал:")) //

{ averagemark = Convert.ToDouble(sr.ReadLine()); }

WRITELINE(path\_1, "\t\tСТУДЕНТ");

WRITELINE(path\_1, "Прізвище:");

WRITELINE(path\_1, surname);

WRITELINE(path\_1, "Ім’я:");

WRITELINE(path\_1, name);

WRITELINE(path\_1, "По-батькові:");

WRITELINE(path\_1, secondname);

WRITELINE(path\_1, "Дата народження:");

WRITELINE(path\_1, datebirthday);

WRITELINE(path\_1, "Рік вступу:");

WRITELINE(path\_1, yearstart);

WRITELINE(path\_1, "Ідентифікаційний код:");

WRITELINE(path\_1, kod.ToString());

WRITELINE(path\_1, "Факультет:");

WRITELINE(path\_1, facultyName);

WRITELINE(path\_1, "Курс:");

WRITELINE(path\_1, course.ToString());

WRITELINE(path\_1, "Назва групи:");

WRITELINE(path\_1, groupname);

WRITELINE(path\_1, "Середній бал:");

WRITELINE(path\_1, averagemark.ToString());

}

if (line == "ВИКЛАДАЧ")//если это преподователь то считай его поля

{

for(int I=1;I<=18;I++)

{ line = sr.ReadLine();}

}

line = sr.ReadLine();

}while(!oznaka1);

return true;

}

}

public bool ReadLECTURERS\_and\_Write\_FromFile()//&&&&&&&&&

{

using (StreamReader sr = new StreamReader(path\_, System.Text.Encoding.GetEncoding(1251)))

{

string[] lines = { };

if (File.Exists(path\_1))

{

File.WriteAllLines(path\_1, lines);

}

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", chair = "//", position = "//";

string facultyName = "";

uint kod = 1000000000;

bool oznaka1 = false;

line = sr.ReadLine();

if (line == null)

{

return false;//если сразу пусто то фолс

}

do

{

if (line == null)//если дошел до конца файла то верни количество ПОСЛЕ

{

oznaka1 = true;

return true;

}

if (line == "СТУДЕНТ")//если это преподователь то считай его поля

{

for(int I=1;I<=20;I++)

{line = sr.ReadLine();}

}

if (line == "ВИКЛАДАЧ")//если это преподователь то считай его поля

{

line = sr.ReadLine();

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік початку роботи:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Кафедра:")) //

{ chair = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Посада:")) //

{ position = sr.ReadLine(); }

WRITELINE(path\_1, "\t\tВИКЛАДАЧ");

WRITELINE(path\_1, "Прізвище:");

WRITELINE(path\_1, surname);

WRITELINE(path\_1, "Ім’я:");

WRITELINE(path\_1, name);

WRITELINE(path\_1, "По-батькові:");

WRITELINE(path\_1, secondname);

//WRITELINE(path\_1, "Дата народження:");

//WRITELINE(path\_1, datebirthday);

WRITELINE(path\_1, "Рік початку роботи:");

WRITELINE(path\_1, yearstart);

//WRITELINE(path\_1, "Ідентифікаційний код:");

//WRITELINE(path\_1, kod.ToString());

WRITELINE(path\_1, "Факультет:");

WRITELINE(path\_1, facultyName);

WRITELINE(path\_1, "Кафедра:");

WRITELINE(path\_1, chair);

WRITELINE(path\_1, "Посада:");

WRITELINE(path\_1, position);

}

line = sr.ReadLine();

}while(!oznaka1);

return true;

}

}

void HiddenAll()

{

resulting1.Visibility = Visibility.Hidden;

resulting2.Visibility = Visibility.Hidden;

resulting3.Visibility = Visibility.Hidden;

resulting5.Visibility = Visibility.Hidden;

resulting6.Visibility = Visibility.Hidden;

resulting7.Visibility = Visibility.Hidden;

resulting8.Visibility = Visibility.Hidden;

resulting10.Visibility = Visibility.Hidden;

resulting11.Visibility = Visibility.Hidden;

SortMenu.Visibility = Visibility.Hidden;

SortButton.Visibility = Visibility.Hidden;

SortLabel1.Visibility = Visibility.Hidden;

SearchLabel.Visibility = Visibility.Hidden;

Menu.Visibility = Visibility.Hidden;

FoundButton1.Visibility = Visibility.Hidden;

FoundButton2.Visibility = Visibility.Hidden;

Found1.Visibility = Visibility.Hidden;

Found2.Visibility = Visibility.Hidden;

SortLabel.Visibility = Visibility.Hidden;

SortMenu.Visibility = Visibility.Hidden;

SurnameLabel.Visibility = Visibility.Hidden;

NameLabel.Visibility = Visibility.Hidden;

FatherNameLabel.Visibility = Visibility.Hidden;

ChaireLabel.Visibility = Visibility.Hidden;

PositionLabel.Visibility = Visibility.Hidden;

YearStartLabel.Visibility = Visibility.Hidden;

SurnameText.Visibility = Visibility.Hidden;

NameText.Visibility = Visibility.Hidden;

FatherNameText.Visibility = Visibility.Hidden;

ChaireText.Visibility = Visibility.Hidden;

PositionText.Visibility = Visibility.Hidden;

FacultiesLabel.Visibility = Visibility.Hidden;

YearStartText.Visibility = Visibility.Hidden;

GroupNameLabel.Visibility = Visibility.Hidden;

GroupNameText.Visibility = Visibility.Hidden;

GroupNameLabel.Visibility = Visibility.Hidden;

GroupNameText.Visibility = Visibility.Hidden;

CourseLabel.Visibility = Visibility.Hidden;

CourseText.Visibility = Visibility.Hidden;

FacultyText.Visibility = Visibility.Hidden;

FacultiesLabel.Visibility = Visibility.Hidden;

resulting1.Content = "";

resulting2.Content = "";

resulting3.Content = "";

resulting5.Content = "";

resulting6.Content = "";

resulting7.Content = "";

resulting8.Content = "";

resulting10.Content = "";

resulting11.Content = "";

SurnameText.Text = "";

NameText.Text = "";

FatherNameText.Text = "";

ChaireText.Text = "";

PositionText.Text = "";

YearStartText.Text = "";

GroupNameText.Text = "";

CourseText.Text = "";

FacultyText.Text = "";

Menu.SelectedIndex = -1;

}

void Visible()

{

resulting1.Visibility = Visibility.Hidden;

resulting2.Visibility = Visibility.Hidden;

resulting3.Visibility = Visibility.Hidden;

resulting5.Visibility = Visibility.Hidden;

resulting6.Visibility = Visibility.Hidden;

resulting7.Visibility = Visibility.Hidden;

resulting8.Visibility = Visibility.Hidden;

resulting10.Visibility = Visibility.Hidden;

resulting11.Visibility = Visibility.Hidden;

Menu.Visibility = Visibility.Visible;

FoundButton1.Visibility = Visibility.Visible;

FoundButton2.Visibility = Visibility.Visible;

Found1.Visibility = Visibility.Visible;

Found2.Visibility = Visibility.Visible;

SurnameLabel.Visibility = Visibility.Visible;

NameLabel.Visibility = Visibility.Visible;

FatherNameLabel.Visibility = Visibility.Visible;

YearStartLabel.Visibility = Visibility.Visible;

SurnameText.Visibility = Visibility.Visible;

NameText.Visibility = Visibility.Visible;

FatherNameText.Visibility = Visibility.Visible;

FacultiesLabel.Visibility = Visibility.Visible;

YearStartText.Visibility = Visibility.Visible;

FacultyText.Visibility = Visibility.Visible;

FacultiesLabel.Visibility = Visibility.Visible;

}

void VisibleTeacher()

{

SurnameLabel.Visibility = Visibility.Visible;

NameLabel.Visibility = Visibility.Visible;

FatherNameLabel.Visibility = Visibility.Visible;

ChaireLabel.Visibility = Visibility.Visible;

PositionLabel.Visibility = Visibility.Hidden;

FacultiesLabel.Visibility = Visibility.Hidden;

YearStartLabel.Visibility = Visibility.Visible;

SurnameText.Visibility = Visibility.Visible;

NameText.Visibility = Visibility.Visible;

FatherNameText.Visibility = Visibility.Visible;

ChaireText.Visibility = Visibility.Visible;

PositionText.Visibility = Visibility.Hidden;

YearStartText.Visibility = Visibility.Visible;

GroupNameLabel.Visibility = Visibility.Hidden;

GroupNameText.Visibility = Visibility.Hidden;

CourseLabel.Visibility = Visibility.Hidden;

CourseText.Visibility = Visibility.Hidden;

FacultyText.Visibility = Visibility.Hidden;

FacultiesLabel.Visibility = Visibility.Hidden;

}

void VisibleStudent()

{

SurnameLabel.Visibility = Visibility.Visible;

NameLabel.Visibility = Visibility.Visible;

FatherNameLabel.Visibility = Visibility.Visible;

SurnameText.Visibility = Visibility.Visible;

NameText.Visibility = Visibility.Visible;

FatherNameText.Visibility = Visibility.Visible;

GroupNameLabel.Visibility = Visibility.Visible;

GroupNameText.Visibility = Visibility.Visible;

YearStartLabel.Visibility = Visibility.Visible;

YearStartText.Visibility = Visibility.Visible;

CourseLabel.Visibility = Visibility.Visible;

CourseText.Visibility = Visibility.Visible;

FacultyText.Visibility = Visibility.Visible;

FacultiesLabel.Visibility = Visibility.Visible;

}

private void Button\_Click(object sender, RoutedEventArgs e)//пошук

{

HiddenAll();

Found1.Visibility = Visibility.Visible;

Found2.Visibility = Visibility.Visible;

Menu.Visibility = Visibility.Visible;

FoundButton1.Visibility = Visibility.Visible;

}

private void FoundButton1\_Click(object sender, RoutedEventArgs e)//продовжити

{

Visible();

if (Menu.SelectedIndex == 0) { VisibleTeacher(); }

if (Menu.SelectedIndex == 1) { VisibleStudent(); }

FoundButton2.Visibility = Visibility.Visible;

FoundButton1.Visibility = Visibility.Hidden;

}

private void FoundButton2\_Click(object sender, RoutedEventArgs e)//знайти

{

int choose = Menu.SelectedIndex;

switch (choose)

{

case 0://vukladach

{

VisibleTeacher();

FoundButton2.Visibility = Visibility.Visible;

resulting1.Visibility = Visibility.Visible;

resulting2.Visibility = Visibility.Visible;

resulting3.Visibility = Visibility.Visible;

resulting5.Visibility = Visibility.Visible;

resulting6.Visibility = Visibility.Visible;

if (Menu.SelectedIndex == 0 )//lecturer

{

Regex reg = new Regex(@"[А-ЯЇІЄЙ]([а-яйіїєює]){1,}");

Match m = reg.Match(SurnameText.Text); //Прізвище

if (m.Success)

{

resulting1.Foreground = Brushes.Green;

resulting1.Content = "OK";

}

else

{

resulting1.Foreground = Brushes.Red;

resulting1.Content = "Неправильно";

}

reg = new Regex(@"[А-ЯЇІЄЙ]([а-яйіїєює]){1,}");

m = reg.Match(NameText.Text); //Ім’я

if (m.Success)

{

resulting2.Foreground = Brushes.Green;

resulting2.Content = "OK";

}

else

{

resulting2.Foreground = Brushes.Red;

resulting2.Content = "Неправильно";

}

reg = new Regex(@"[А-ЯЇІЄЙ]([а-яйіїєює]){1,}");

m = reg.Match(FatherNameText.Text); //По батькові

if (m.Success)

{

resulting3.Foreground = Brushes.Green;

resulting3.Content = "OK";

}

else

{

resulting3.Foreground = Brushes.Red;

resulting3.Content = "Неправильно";

}

reg = new Regex(@"(196[2-9]|19[789]\d|200\d|201[0-7])");

m = reg.Match(YearStartText.Text); //Рік старт

if (m.Success)

{

resulting5.Foreground = Brushes.Green;

resulting5.Content = "OK";

}

else

{

resulting5.Foreground = Brushes.Red;

resulting5.Content = "Неправильно";

}

resulting10.Visibility = Visibility.Visible;

resulting11.Visibility = Visibility.Visible;

Regex reg1 = new Regex(@"КТМ|(КА(іАГ)|(тКІТ))|(КФ(тВМ)|(ВіС)|(ФіК))|КЗД|(КГ(М)|(іСН))|(КК(СУтА)|І)|КПЗС|(КР(ДАтТ)|(РКК))|(КМ(ОіА)|Е)|(КЕ(тП)|(БПУтА))|КІМ|КОіА|КМ|КЕ");

Match m1 = reg1.Match(ChaireText.Text); //Кафедра

if (m1.Success)

{

resulting10.Foreground = Brushes.Green;

resulting10.Content = "OK";

}

else

{

resulting10.Foreground = Brushes.Red;

resulting10.Content = "Неправильно";

}

if (ReadLECTURERS\_AndFoundFromFile(NameText.Text, SurnameText.Text, FatherNameText.Text, YearStartText.Text,ChaireText.Text))

{ MessageBox.Show(" Такий ВИКЛАДАЧ існує!\n Переглнути інформацію про нього можна у файлі 'course\_project.txt'", "Повідомлення!", MessageBoxButton.OK, MessageBoxImage.Information); }

}

break;

}

case 1://student

{

HiddenAll();

Found1.Visibility = Visibility.Visible;

Found2.Visibility = Visibility.Visible;

VisibleStudent();

Menu.Visibility = Visibility.Visible;

FoundButton2.Visibility = Visibility.Visible;

resulting1.Visibility = Visibility.Visible;

resulting2.Visibility = Visibility.Visible;

resulting3.Visibility = Visibility.Visible;

resulting5.Visibility = Visibility.Visible;

resulting6.Visibility = Visibility.Visible;

if (Menu.SelectedIndex == 1)//student

{

Regex reg = new Regex(@"[А-ЯЇІЄЙ]([а-яйіїєює]){1,}");

Match m = reg.Match(SurnameText.Text); //Прізвище

if (m.Success)

{

resulting1.Foreground = Brushes.Green;

resulting1.Content = "OK";

}

else

{

resulting1.Foreground = Brushes.Red;

resulting1.Content = "Неправильно";

}

reg = new Regex(@"[А-ЯЇІЄЙ]([а-яйіїєює]){1,}");

m = reg.Match(NameText.Text); //Ім’я

if (m.Success)

{

resulting2.Foreground = Brushes.Green;

resulting2.Content = "OK";

}

else

{

resulting2.Foreground = Brushes.Red;

resulting2.Content = "Неправильно";

}

reg = new Regex(@"[А-ЯЇІЄЙ]([а-яйіїєює]){1,}");

m = reg.Match(FatherNameText.Text); //По батькові

if (m.Success)

{

resulting3.Foreground = Brushes.Green;

resulting3.Content = "OK";

}

else

{

resulting3.Foreground = Brushes.Red;

resulting3.Content = "Неправильно";

}

reg = new Regex(@"(196[2-9]|19[789]\d|200\d|201[0-7])");

m = reg.Match(YearStartText.Text); //Рік старт

if (m.Success)

{

resulting5.Foreground = Brushes.Green;

resulting5.Content = "OK";

}

else

{

resulting5.Foreground = Brushes.Red;

resulting5.Content = "Неправильно";

}

reg = new Regex(@"(Ф((ІМ)|(ІКТ))|Ф((УМ)|(ОФ)))|(ГЕФ)");

m = reg.Match(FacultyText.Text); //Факультет

if (m.Success)

{

resulting6.Foreground = Brushes.Green;

resulting6.Content = "OK";

}

else

{

resulting6.Foreground = Brushes.Red;

resulting6.Content = "Неправильно";

}

resulting8.Visibility = Visibility.Visible;

resulting7.Visibility = Visibility.Visible;

Regex reg1 = new Regex(@"[1-5]");

Match m1 = reg1.Match(CourseText.Text); //Курс

if (m1.Success)

{

resulting7.Foreground = Brushes.Green;

resulting7.Content = "OK";

}

else

{

resulting7.Foreground = Brushes.Red;

resulting7.Content = "Неправильно";

}

reg1 = new Regex(@"[А-ЯІЇЄЙ]{2,6}\-\d{1,3}");

m1 = reg1.Match(GroupNameText.Text); //Назва групи

if (m1.Success)

{

resulting8.Foreground = Brushes.Green;

resulting8.Content = "OK";

}

else

{

resulting8.Foreground = Brushes.Red;

resulting8.Content = "Неправильно";

}

if (ReadSTUDENT\_AndFoundFromFile(NameText.Text, SurnameText.Text, FatherNameText.Text, CourseText.Text, FacultyText.Text, GroupNameText.Text, YearStartText.Text))

{

File.Copy(@"G:\KursovaFleshechna\WPFMain\bin\Debug\course\_project1.txt", @"G:\KursovaFleshechna\course\_project.txt", true);

MessageBox.Show("Такий СТУДЕНТ існує!\n Переглнути інформацію про нього можна у файлі 'course\_project.txt'", "Повідомлення!", MessageBoxButton.OK, MessageBoxImage.Information); }

}

break;

}

//

default: break;

}

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)//сортування

{

HiddenAll();

SortLabel.Visibility = Visibility.Visible;

SortMenu.Visibility = Visibility.Visible;

SortLabel1.Visibility = Visibility.Visible;

SortButton.Visibility = Visibility.Visible;

}

private void Button\_Click\_2(object sender, RoutedEventArgs e)//перегляд

{

HiddenAll();

SearchLabel.Visibility = Visibility.Visible;

ReadSTUDENT\_LECTURERS\_and\_Write\_FromFile();

File.Copy(@"G:\KursovaFleshechna\WPFMain\bin\Debug\course\_project1.txt", @"G:\KursovaFleshechna\course\_project.txt", true);

}

private void SortButton\_Click(object sender, RoutedEventArgs e)//кнопка сортувати

{

int choose = SortMenu.SelectedIndex;

switch (choose)

{

case 0://Викладачі

{

ReadLECTURERS\_and\_Write\_FromFile();

File.Copy(@"G:\KursovaFleshechna\WPFMain\bin\Debug\course\_project1.txt", @"G:\KursovaFleshechna\course\_project.txt", true);

break;

}

case 1://Студенти

{

ReadSTUDENT\_and\_Write\_FromFile();

File.Copy(@"G:\KursovaFleshechna\WPFMain\bin\Debug\course\_project1.txt", @"G:\KursovaFleshechna\course\_project.txt", true);

break;

}

default: break;

}

}

}

}

AdminOpportunity.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using System.Text.RegularExpressions;

using System.Web.RegularExpressions;

using LibraryForKursovaya;

using System.IO;

namespace WPFMain

{

/// <summary>

/// Interaction logic for AdminOpportunity.xaml

/// </summary>

public partial class AdminOpportunity : Window

{

public string path\_ = @"course\_project.txt";

public string path\_1 = @"course\_project1.txt";

public Dictionary<uint,Student> Add\_And\_Print\_PointSTUDENTToDictionary(string pyt, string group,int course, double averageMark, string name, string surName,string secondName, uint iD\_Kod, string dateBirthday, string yearStart, string facultyName)

{

Dictionary<uint,Student> dict = new Dictionary<uint,Student>();

Student student=new Student(name,surName,secondName,iD\_Kod,dateBirthday,yearStart,facultyName,group,course,averageMark);

dict.Add(iD\_Kod, student);

using (StreamWriter sw = new StreamWriter(pyt, true, System.Text.Encoding.GetEncoding(1251)))

{

sw.WriteLine("СТУДЕНТ");

sw.WriteLine("Прізвище:");

sw.WriteLine(surName);

sw.WriteLine("Ім’я:");

sw.WriteLine(name);

sw.WriteLine("По-батькові:");//path\_

sw.WriteLine(secondName);

sw.WriteLine("Дата народження:");

sw.WriteLine(dateBirthday);

sw.WriteLine("Рік вступу:");

sw.WriteLine(yearStart);

sw.WriteLine("Ідентифікаційний код:");

sw.WriteLine(iD\_Kod);

sw.WriteLine("Факультет:");

sw.WriteLine(facultyName);

sw.WriteLine("Курс:");

sw.WriteLine(course);

sw.WriteLine("Назва групи:");

sw.WriteLine(group);

sw.WriteLine("Середній бал:");

sw.WriteLine(averageMark);

}

return dict;

}//+++++++++

public Dictionary<uint, Lecturer> Add\_And\_Print\_PointLECTURERToDictionary(string pyt, string chair, string position, string name, string surName, string secondName, uint iD\_Kod, string dateBirthday, string yearStart, string facultyName)

{

Dictionary<uint, Lecturer> dict = new Dictionary<uint, Lecturer>();

Lecturer lecturer = new Lecturer(name, surName, secondName, iD\_Kod, dateBirthday, yearStart, facultyName, chair, position);

dict.Add(iD\_Kod, lecturer);

using (StreamWriter sw = new StreamWriter(pyt, true, System.Text.Encoding.Default))

{

sw.WriteLine("ВИКЛАДАЧ");

sw.WriteLine("Прізвище:");

sw.WriteLine(surName);

sw.WriteLine("Ім’я:");

sw.WriteLine(name);

sw.WriteLine("По-батькові:");

sw.WriteLine(secondName);

sw.WriteLine("Дата народження:");

sw.WriteLine(dateBirthday);

sw.WriteLine("Рік початку роботи:");

sw.WriteLine(yearStart);

sw.WriteLine("Ідентифікаційний код:");

sw.WriteLine(iD\_Kod);

sw.WriteLine("Факультет:");

sw.WriteLine(facultyName);

sw.WriteLine("Кафедра:");

sw.WriteLine(chair);

sw.WriteLine("Посада:");

sw.WriteLine(position);

}

return dict;

}//+++++++++

public bool DeleteSTUDENTFromFile(uint kod)

{

if (ExistSTUDENT\_InFile(kod))//если ключ у єтого студента такой то удалить

{

int c = ReadSTUDENT\_and\_Count\_FromFile(kod);//95

int c1 = ReadSTUDENT\_and\_Count\_FromFile2(kod);//139

bool deleting = ReadSTUDENT\_and\_Count\_and\_Write\_FromFile(c, c1);

return true;

}

else

return false; //если такого нету то и не удалит его

}

public bool DeleteLECTURERFromFile(uint kod)

{

if (ExistLECTURER\_InFile(kod))//если ключ у єтого prepodovatelia такой то дальше

{

int c= ReadLECTURER\_and\_Count\_FromFile(kod);//38

int c1 = ReadLECTURER\_and\_Count\_FromFile2(kod);//139

bool deleting=ReadLECTURER\_and\_Count\_and\_Write\_FromFile(c,c1);

return true;

}

else

return false; //если такого нету то и не удалит его

}

public int ReadSTUDENT\_and\_Count\_FromFile(uint kod1)//111111111111111111111&

{

using (StreamReader sr = new StreamReader(path\_, System.Text.Encoding.Default))

{

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", groupname = "//";

int course = -1;

int counter = 0;

double averagemark = 1.0;

string facultyName = "";

uint kod = 1000000000;

bool oznaka1 = false;

line = sr.ReadLine();

if (line == null)

{

return 0;

}

do

{

if (line == null)

{

return 0;

}

if (line == "СТУДЕНТ")

{

line = sr.ReadLine();

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік вступу:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Курс:")) //

{ course = Convert.ToUInt16(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Назва групи:")) //

{ groupname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Середній бал:")) //

{ averagemark = Convert.ToDouble(sr.ReadLine()); }

if (kod1 != kod)

{

counter += 21;

}

if (kod1 == kod)

{

oznaka1 = true;

}

}

else

{

if (line == "ВИКЛАДАЧ")

{

for (int I = 1; I <= 18; I++)

{ line = sr.ReadLine(); }

counter += 19;

}

}

line = sr.ReadLine();

} while (!oznaka1);

return counter;

}

}

public void WRITELINE(string pyt, string word)

{

using (StreamWriter sw = new StreamWriter(pyt, true, System.Text.Encoding.GetEncoding(1251)))

{

sw.WriteLine(word);

}

}

public int ReadLECTURER\_and\_Count\_FromFile(uint kod1)//111111111111111111111&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{

using (StreamReader sr = new StreamReader(path\_, System.Text.Encoding.Default))

{

int counter = 0;

uint kod = 0;

bool oznaka1 = false;

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", chair = "//", position = "//";

string facultyName = "ФІКТ";

line = sr.ReadLine();

if (line == null)

{

return 0;

}

do

{

if (line == null)

{

return 0;

}

if (line == "ВИКЛАДАЧ")

{

line = sr.ReadLine();

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік початку роботи:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Кафедра:")) //

{ chair = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Посада:")) //

{ position = sr.ReadLine(); }

if (kod1 != kod)

{

counter += 19;

}

if (kod1 == kod)

{

oznaka1 = true;

}

}

else

{

if (line == "СТУДЕНТ")

{

for (int I = 1; I <= 20; I++)

{ line = sr.ReadLine(); }

counter += 21;

}

}

line = sr.ReadLine();

} while (!oznaka1);

return counter;

}

}

public int ReadLECTURER\_and\_Count\_FromFile2(uint kod1)//222222222222222222222222

{

using (StreamReader sr = new StreamReader(path\_, System.Text.Encoding.Default))

{

int counter = 0;

uint kod = 0;

bool oznaka1 = false;

bool oznaka2 = false;

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", chair = "//", position = "//";

string facultyName = "ФІКТ";

line = sr.ReadLine();

if (line == null)

{

return 0;

}

do

{

if (line == "ВИКЛАДАЧ" && (line = sr.ReadLine()) != null)

{

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік початку роботи:"))

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Кафедра:")) //

{ chair = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Посада:")) //

{ position = sr.ReadLine(); }

if (kod1 != kod && oznaka1 == true)

{

counter += 19;

}

if (kod1 == kod)

{

oznaka1 = true;

}

}

else

{

if (line == "СТУДЕНТ" && oznaka1 == true)

{

for (int I = 1; I <= 20; I++)

{ line = sr.ReadLine(); }

counter += 21;

}

}

line = sr.ReadLine();

if (line == null) { return counter; }

} while (!oznaka2);

return counter;

}

}

public bool ReadLECTURER\_and\_Count\_and\_Write\_FromFile(int Counter, int Counter1)//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{

using (StreamReader sr = new StreamReader(path\_, System.Text.Encoding.Default))

{

int counter = 0, counter1 = 0;

uint kod = 0;

bool oznaka1 = false;

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", chair = "//", position = "//";

string facultyName = "ФІКТ";

line = sr.ReadLine();

if (line == null)

{

return false;//если сразу пусто то фолс

}

do

{

if (line == null)//если дошел до конца файла то верни количество ПОСЛЕ

{

oznaka1 = true;

return true;

}

if (line == "ВИКЛАДАЧ")//если это преподователь то считай его поля

{

line = sr.ReadLine();

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік початку роботи:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Кафедра:")) //

{ chair = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Посада:")) //

{ position = sr.ReadLine(); }

counter += 19;//добавь количество строк для преподователя

//if (counter < Counter)//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!111

//{

WRITELINE(path\_1, "ВИКЛАДАЧ");

WRITELINE(path\_1, "Прізвище:");

WRITELINE(path\_1, surname);

WRITELINE(path\_1, "Ім’я:");

WRITELINE(path\_1, name);

WRITELINE(path\_1, "По-батькові:");

WRITELINE(path\_1, secondname);

WRITELINE(path\_1, "Дата народження:");

WRITELINE(path\_1, datebirthday);

WRITELINE(path\_1, "Рік початку роботи:");

WRITELINE(path\_1, yearstart);

WRITELINE(path\_1, "Ідентифікаційний код:");

WRITELINE(path\_1, kod.ToString());

WRITELINE(path\_1, "Факультет:");

WRITELINE(path\_1, facultyName);

WRITELINE(path\_1, "Кафедра:");

WRITELINE(path\_1, chair);

WRITELINE(path\_1, "Посада:");

WRITELINE(path\_1, position);

if (counter == Counter)//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

{

for (int I = 1; I <= 19; I++)//если ты попал на преподователя которого нужно удалить то пройди все его строки не копируя в новый файл

{ line = sr.ReadLine(); }

bool oznaka2 = false;

do

{

if (((line = sr.ReadLine()) != null) || Counter1 != 0)

{

if (line == "ВИКЛАДАЧ")//если это преподователь то считай его поля

{

line = sr.ReadLine();

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік початку роботи:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Кафедра:")) //

{ chair = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Посада:")) //

{ position = sr.ReadLine(); }

counter1 += 19;//добавь количество строк для преподователя

WRITELINE(path\_1, "ВИКЛАДАЧ");

WRITELINE(path\_1, "Прізвище:");

WRITELINE(path\_1, surname);

WRITELINE(path\_1, "Ім’я:");

WRITELINE(path\_1, name);

WRITELINE(path\_1, "По-батькові:");

WRITELINE(path\_1, secondname);

WRITELINE(path\_1, "Дата народження:");

WRITELINE(path\_1, datebirthday);

WRITELINE(path\_1, "Рік початку роботи:");

WRITELINE(path\_1, yearstart);

WRITELINE(path\_1, "Ідентифікаційний код:");

WRITELINE(path\_1, kod.ToString());

WRITELINE(path\_1, "Факультет:");

WRITELINE(path\_1, facultyName);

WRITELINE(path\_1, "Кафедра:");

WRITELINE(path\_1, chair);

WRITELINE(path\_1, "Посада:");

WRITELINE(path\_1, position);

if (counter1 == Counter1)

{

oznaka2 = true;

return true;

}

}

else

{

if (line == "СТУДЕНТ")//если это студент то запиши все его поля

{

WRITELINE(path\_1, "СТУДЕНТ");

for (int I = 1; I <= 20; I++)

{

line = sr.ReadLine();

WRITELINE(path\_1, line);

}

counter1 += 21;//добавь количество всех полей студента

}

if (counter1 == Counter1)

{

oznaka1 = true;

return true;

}

}

}

else return true;

} while (!oznaka2);

}

}

else

{

if (line == "СТУДЕНТ")//если это студент то запиши все его поля

{//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

WRITELINE(path\_1, "СТУДЕНТ");

for (int I = 1; I <= 20; I++)

{

line = sr.ReadLine();

WRITELINE(path\_1, line);

}

counter += 21;//добавь количество всех полей студента

}

if (counter == Counter)

{

for (int I = 1; I <= 19; I++)//если ты попал на преподователя которого нужно удалить то пройди все его строки не копируя в новый файл

{ line = sr.ReadLine(); }

bool oznaka2 = false;

do

{

if (((line = sr.ReadLine()) != null) && Counter1 != 0)

{

if (line == "ВИКЛАДАЧ")//если это преподователь то считай его поля

{

line = sr.ReadLine();

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік початку роботи:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Кафедра:")) //

{ chair = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Посада:")) //

{ position = sr.ReadLine(); }

counter1 += 19;//добавь количество строк для преподователя

WRITELINE(path\_1, "ВИКЛАДАЧ");

WRITELINE(path\_1, "Прізвище:");

WRITELINE(path\_1, surname);

WRITELINE(path\_1, "Ім’я:");

WRITELINE(path\_1, name);

WRITELINE(path\_1, "По-батькові:");

WRITELINE(path\_1, secondname);

WRITELINE(path\_1, "Дата народження:");

WRITELINE(path\_1, datebirthday);

WRITELINE(path\_1, "Рік початку роботи:");

WRITELINE(path\_1, yearstart);

WRITELINE(path\_1, "Ідентифікаційний код:");

WRITELINE(path\_1, kod.ToString());

WRITELINE(path\_1, "Факультет:");

WRITELINE(path\_1, facultyName);

WRITELINE(path\_1, "Кафедра:");

WRITELINE(path\_1, chair);

WRITELINE(path\_1, "Посада:");

WRITELINE(path\_1, position);

if (counter1 == Counter1)

{

return true;

}

}

else

{

if (line == "СТУДЕНТ")//если это студент то запиши все его поля

{

WRITELINE(path\_1, "СТУДЕНТ");

for (int I = 1; I <= 20; I++)

{

line = sr.ReadLine();

WRITELINE(path\_1, line);

}

counter1 += 21;//добавь количество всех полей студента

}

if (counter1 == Counter1)

{

oznaka1 = true;

return true;

}

}

}

else return true;

} while (!oznaka2);

}

}

line = sr.ReadLine();//

} while (!oznaka1);

return oznaka1;

}

}

public bool ReadSTUDENT\_and\_Count\_and\_Write\_FromFile(int Counter, int Counter1)//&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

{

using (StreamReader sr = new StreamReader(path\_, System.Text.Encoding.Default))

{

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", groupname = "//";

int course = -1;

int counter = 0;

int counter1 = 0;

double averagemark = 1.0;

string facultyName = "";

uint kod = 1000000000;

bool oznaka1 = false;

line = sr.ReadLine();

if (line == null)

{

return false;//если сразу пусто то фолс

}

do

{

if (line == null)//если дошел до конца файла то верни количество ПОСЛЕ

{

oznaka1 = true;

return true;

}

if (line == "СТУДЕНТ")//если это преподователь то считай его поля

{

line = sr.ReadLine();

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік вступу:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Курс:")) //

{ course = Convert.ToUInt16(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Назва групи:")) //

{ groupname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Середній бал:")) //

{ averagemark = Convert.ToDouble(sr.ReadLine()); }

counter += 21;//добавь количество строк для преподователя

WRITELINE(path\_1, "СТУДЕНТ");

WRITELINE(path\_1, "Прізвище:");

WRITELINE(path\_1, surname);

WRITELINE(path\_1, "Ім’я:");

WRITELINE(path\_1, name);

WRITELINE(path\_1, "По-батькові:");

WRITELINE(path\_1, secondname);

WRITELINE(path\_1, "Дата народження:");

WRITELINE(path\_1, datebirthday);

WRITELINE(path\_1, "Рік вступу:");

WRITELINE(path\_1, yearstart);

WRITELINE(path\_1, "Ідентифікаційний код:");

WRITELINE(path\_1, kod.ToString());

WRITELINE(path\_1, "Факультет:");

WRITELINE(path\_1, facultyName);

WRITELINE(path\_1, "Курс:");

WRITELINE(path\_1, course.ToString());

WRITELINE(path\_1, "Назва групи:");

WRITELINE(path\_1, groupname);

WRITELINE(path\_1, "Середній бал:");

WRITELINE(path\_1, averagemark.ToString());

if (counter == Counter)//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

{

for (int I = 1; I <= 21; I++)//если ты попал на преподователя которого нужно удалить то пройди все его строки не копируя в новый файл

{ line = sr.ReadLine(); }

bool oznaka2 = false;

do

{

if (((line = sr.ReadLine()) != null) || Counter1 != 0)

{

if (line == "СТУДЕНТ")//если это преподователь то считай его поля

{

line = sr.ReadLine();

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік вступу:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Курс:")) //

{ course = Convert.ToUInt16(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Назва групи:")) //

{ groupname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Середній бал:")) //

{ averagemark = Convert.ToDouble(sr.ReadLine()); }

counter1 += 21;//добавь количество строк для преподователя

WRITELINE(path\_1, "СТУДЕНТ");

WRITELINE(path\_1, "Прізвище:");

WRITELINE(path\_1, surname);

WRITELINE(path\_1, "Ім’я:");

WRITELINE(path\_1, name);

WRITELINE(path\_1, "По-батькові:");

WRITELINE(path\_1, secondname);

WRITELINE(path\_1, "Дата народження:");

WRITELINE(path\_1, datebirthday);

WRITELINE(path\_1, "Рік вступу:");

WRITELINE(path\_1, yearstart);

WRITELINE(path\_1, "Ідентифікаційний код:");

WRITELINE(path\_1, kod.ToString());

WRITELINE(path\_1, "Факультет:");

WRITELINE(path\_1, facultyName);

WRITELINE(path\_1, "Курс:");

WRITELINE(path\_1, course.ToString());

WRITELINE(path\_1, "Назва групи:");

WRITELINE(path\_1, groupname);

WRITELINE(path\_1, "Середній бал:");

WRITELINE(path\_1, averagemark.ToString());

if (counter1 == Counter1)

{

oznaka2 = true;

return true;

}

}

else

{

if (line == "ВИКЛАДАЧ")//если это студент то запиши все его поля

{

WRITELINE(path\_1, "ВИКЛАДАЧ");

for (int I = 1; I <= 19; I++)

{

line = sr.ReadLine();

WRITELINE(path\_1, line);

}

counter1 += 19;//добавь количество всех полей студента

}

if (counter1 == Counter1)

{

oznaka1 = true;

return true;

}

}

}

else return true;

if (line == null) { oznaka2 = true; }

} while (!oznaka2);

}

}

else

{

if (line == "ВИКЛАДАЧ")//если это студент то запиши все его поля

{

WRITELINE(path\_1, "ВИКЛАДАЧ");

for (int I = 1; I <= 18; I++)

{

line = sr.ReadLine();

WRITELINE(path\_1, line);

}

counter += 19;//добавь количество всех полей студента

}

if (counter == Counter)

{

for (int I = 1; I <= 21; I++)//если ты попал на преподователя которого нужно удалить то пройди все его строки не копируя в новый файл

{ line = sr.ReadLine(); }

bool oznaka2 = false;

do

{

if (((line = sr.ReadLine()) != null) && Counter1 != 0)

{

if (line == "СТУДЕНТ")//если это преподователь то считай его поля

{

line = sr.ReadLine();

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік вступу:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Курс:")) //

{ course = Convert.ToUInt16(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Назва групи:")) //

{ groupname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Середній бал:")) //

{ averagemark = Convert.ToDouble(sr.ReadLine()); }

counter1 += 21;//добавь количество строк для преподователя

WRITELINE(path\_1, "СТУДЕНТ");

WRITELINE(path\_1, "Прізвище:");

WRITELINE(path\_1, surname);

WRITELINE(path\_1, "Ім’я:");

WRITELINE(path\_1, name);

WRITELINE(path\_1, "По-батькові:");

WRITELINE(path\_1, secondname);

WRITELINE(path\_1, "Дата народження:");

WRITELINE(path\_1, datebirthday);

WRITELINE(path\_1, "Рік вступу:");

WRITELINE(path\_1, yearstart);

WRITELINE(path\_1, "Ідентифікаційний код:");

WRITELINE(path\_1, kod.ToString());

WRITELINE(path\_1, "Факультет:");

WRITELINE(path\_1, facultyName);

WRITELINE(path\_1, "Курс:");

WRITELINE(path\_1, course.ToString());

WRITELINE(path\_1, "Назва групи:");

WRITELINE(path\_1, groupname);

WRITELINE(path\_1, "Середній бал:");

WRITELINE(path\_1, averagemark.ToString());

if (counter1 == Counter1)

{

oznaka2 = true;

return true;

}

}

else

{

if (line == "ВИКЛАДАЧ")//если это студент то запиши все его поля

{

WRITELINE(path\_1, "ВИКЛАДАЧ");

for (int I = 1; I <= 18; I++)

{

line = sr.ReadLine();

WRITELINE(path\_1, line);

}

counter1 += 19;//добавь количество всех полей студента

}

if (counter1 == Counter1)

{

oznaka1 = true;

return true;

}

}

}

else return true;

} while (!oznaka2);

}

}

line = sr.ReadLine();//

if (line == null)

{ oznaka1 = true; }

} while (!oznaka1);

return oznaka1;

}

}

public bool ExistSTUDENT\_InFile(uint kod)

{

Dictionary<uint, Student> dict = ReadSTUDENT\_AndFoundFromFile(kod);

if (dict.ContainsKey(kod) && (kod != 1000000000))//если ключ у єтого studenta такой

{

MessageBox.Show("СТУДЕНТ з таким ідентифікаційним кодом існує!\n Отже його запис можна видалити, але неможливо додати іншого з тим самим кодом", "Повідомлення!", MessageBoxButton.OK, MessageBoxImage.Information);

return true; //yes,exist

}

return false; //если такого нету то и не удалит его

}

public bool ExistLECTURER\_InFile(uint kod)

{

Dictionary<uint, Lecturer> dict = ReadLECTURER\_AndFoundFromFile(kod);

if (dict.ContainsKey(kod)&&(kod!=1000000000))//если ключ у єтого lecturer такой

{

MessageBox.Show("ВИКЛАДАЧ з таким ідентифікаційним кодом існує!\n Отже його запис можна видалити, але неможливо додати іншого з тим самим кодом", "Повідомлення!", MessageBoxButton.OK, MessageBoxImage.Information);

return true; //если такого нету то и не удалит его

}

else return false;

}

public int ReadSTUDENT\_and\_Count\_FromFile2(uint kod1)//222222222222222222222222

{

using (StreamReader sr = new StreamReader(path\_, System.Text.Encoding.Default))

{

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", groupname = "//";

int course = -1;

int counter = 0;

double averagemark = 1.0;

string facultyName = "";

uint kod = 1000000000;

bool oznaka2 = false, oznaka3=false;

line = sr.ReadLine();

if (line == null)

{

oznaka3 = true;

return 0;

}

do

{

if (line == "СТУДЕНТ" && (line = sr.ReadLine()) != null)

{

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік вступу:"))

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Курс:")) //

{ course = Convert.ToUInt16(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Назва групи:")) //

{ groupname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Середній бал:")) //

{ averagemark = Convert.ToDouble(sr.ReadLine()); }

if (kod1 != kod && oznaka2 == true)

{

counter += 21;

}

if (kod1 == kod)

{

oznaka2 = true;

}

}

else

{

if (line == "ВИКЛАДАЧ" && oznaka2 == true)

{

for (int I = 1; I <= 18; I++)

{ line = sr.ReadLine(); }

counter += 19;

}

}

line = sr.ReadLine();

if (line == null)

{

oznaka3 = true;

return counter;

}

} while (!oznaka3);

return counter;

}

}

public Dictionary<uint, Student> ReadSTUDENT\_AndFoundFromFile(uint kod1)

{

using (StreamReader sr = new StreamReader(path\_, System.Text.Encoding.Default))

{

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", groupname = "//";

int course = -1;

double averagemark = 1.0;

string facultyName = "";

uint kod = 1000000000;

bool oznaka1 = false;

Dictionary<uint, Student> dict2 = new Dictionary<uint, Student>();

line = sr.ReadLine();

if (line == null)

{

Student student2 = new Student(name, surname, secondname, kod, datebirthday, yearstart, facultyName, groupname, course, averagemark);

dict2.Add(000000000, student2);

return dict2;

}

do{

if (line == null)

{

Student student2 = new Student(name, surname, secondname, kod, datebirthday, yearstart, facultyName, groupname, course, averagemark);

dict2.Add(1000000000, student2);

oznaka1 = true;

}

if (line == "СТУДЕНТ" && ((line = sr.ReadLine()) != null))

{

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік вступу:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Курс:")) //

{ course = Convert.ToUInt16(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Назва групи:")) //

{ groupname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Середній бал:")) //

{ averagemark = Convert.ToDouble(sr.ReadLine()); }

if (kod1 == kod)

{

Dictionary<uint, Student> dict = new Dictionary<uint, Student>();

Student student1 = new Student(name, surname, secondname, kod, datebirthday, yearstart, facultyName, groupname, course, averagemark);

dict.Add(kod1, student1);

return dict;

}

}

else

{

if (line == "ВИКЛАДАЧ")

{

for (int I = 1; I <= 18; I++)

{ line = sr.ReadLine(); }

}

}

line = sr.ReadLine();

} while (!oznaka1);

return dict2;

}

}

public Dictionary<uint, Lecturer> ReadLECTURER\_AndFoundFromFile(uint kod1)

{

using (StreamReader sr = new StreamReader(path\_, System.Text.Encoding.Default))

{

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", chair = "//", position = "//";

string facultyName = "ФІКТ";

uint kod = 1000000000;

bool oznaka1 = false;

Dictionary<uint, Lecturer> dict2 = new Dictionary<uint, Lecturer>();

line = sr.ReadLine();

if (line == null)

{

Lecturer lecturer2 = new Lecturer(name, surname, secondname, kod, datebirthday, yearstart, facultyName, chair, position);

dict2.Add(1000000000, lecturer2);

return dict2;

}

do{

if (line == null)

{

Lecturer lecturer2 = new Lecturer(name, surname, secondname, kod, datebirthday, yearstart, facultyName, chair, position);

dict2.Add(1000000000, lecturer2);

oznaka1 = true;

}

if (line == "ВИКЛАДАЧ" && (line = sr.ReadLine()) != null)

{

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік початку роботи:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Кафедра:")) //

{ chair = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Посада:")) //

{ position = sr.ReadLine(); }

if (kod1 == kod)

{

Dictionary<uint, Lecturer> dict = new Dictionary<uint, Lecturer>();

Lecturer lecturer1 = new Lecturer(name, surname, secondname, kod, datebirthday, yearstart, facultyName, chair, position);

dict.Add(kod, lecturer1);

return dict;

}

}

else

{

if (line == "СТУДЕНТ")

{

for (int I = 1; I <= 20; I++)

{ line = sr.ReadLine(); }

}

}

line = sr.ReadLine();

}while(!oznaka1);

return dict2;

}

}

public Dictionary<uint, Student> ReadSTUDENTFromFile()

{

using (StreamReader sr = new StreamReader(path\_, System.Text.Encoding.Default))

{

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", groupname = "//";

int course=-1;

double averagemark=-1.0;

string facultyName="ФІКТ" ;

uint kod=1000000000;

line = sr.ReadLine();

if (line == "СТУДЕНТ")

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

{

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname= sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік вступу:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Курс:")) //

{ course = Convert.ToUInt16(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Назва групи:")) //

{ groupname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Середній бал:")) //

{ averagemark = Convert.ToDouble(sr.ReadLine()); }

Dictionary<uint,Student> dict = new Dictionary<uint,Student>();

Student student1 = new Student(name, surname, secondname, kod, datebirthday, yearstart, facultyName, groupname, course, averagemark);

dict.Add(kod, student1);

return dict;

}

Dictionary<uint,Student> dict1= new Dictionary<uint,Student>();

return dict1;

}

}//+++++++++

public Dictionary<uint, Lecturer> ReadLECTURERromFile()

{

using (StreamReader sr = new StreamReader(path\_, System.Text.Encoding.Default))

{

string line, line2, name = "//", surname = "//", secondname = "//", datebirthday = "//", yearstart = "//", chair = "//", position = "//";

string facultyName = "ФІКТ";

uint kod = 1000000000;

line = sr.ReadLine();

if (line == "ВИКЛАДАЧ")

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

{

if ((line == "Прізвище:") && (line2 = sr.ReadLine()) != null) //+

{ surname = line2; }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ім’я:")) //+

{ name = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "По-батькові:")) //

{ secondname = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Дата народження:")) //

{ datebirthday = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Рік початку роботи:")) //

{ yearstart = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Ідентифікаційний код:")) //

{ kod = Convert.ToUInt32(sr.ReadLine()); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Факультет:")) //+

{ facultyName = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Кафедра:")) //

{ chair = sr.ReadLine(); }

if (((line2 = sr.ReadLine()) != null) && (line2 == "Посада:")) //

{ position = sr.ReadLine(); }

Dictionary<uint, Lecturer> dict = new Dictionary<uint, Lecturer>();

Lecturer lecturer1 = new Lecturer(name, surname, secondname, kod, datebirthday, yearstart, facultyName, chair, position);

dict.Add(kod, lecturer1);

return dict;

}

Dictionary<uint, Lecturer> dict1 = new Dictionary<uint, Lecturer>();

return dict1;

}

}//++++++++++

void HiddenAll()

{

HelpForLecturer.Visibility = Visibility.Hidden;

HelpForStudent.Visibility = Visibility.Hidden;

resulting1.Visibility = Visibility.Hidden;

resulting2.Visibility = Visibility.Hidden;

resulting3.Visibility = Visibility.Hidden;

resulting4.Visibility = Visibility.Hidden;

resulting5.Visibility = Visibility.Hidden;

resulting6.Visibility = Visibility.Hidden;

resulting7.Visibility = Visibility.Hidden;

resulting8.Visibility = Visibility.Hidden;

resulting9.Visibility = Visibility.Hidden;

resulting10.Visibility = Visibility.Hidden;

resulting11.Visibility = Visibility.Hidden;

resulting12.Visibility = Visibility.Hidden;

AddButton.Visibility = Visibility.Hidden;

AddButton2.Visibility = Visibility.Hidden;

AddFinishButton.Visibility = Visibility.Hidden;

ChangeButton.Visibility = Visibility.Hidden;

ChangeFinishButton.Visibility = Visibility.Hidden;

DeleteButton.Visibility = Visibility.Hidden;

DeleteLabel1.Visibility = Visibility.Hidden;

DeleteLabel2.Visibility = Visibility.Hidden;

DeleteText.Visibility = Visibility.Hidden;

DeleteLabel4.Visibility = Visibility.Hidden;

Menu.Visibility = Visibility.Hidden;

Menu1.Visibility = Visibility.Hidden;

Menu3.Visibility = Visibility.Hidden;

SurnameLabel.Visibility = Visibility.Hidden;

NameLabel.Visibility = Visibility.Hidden;

FatherNameLabel.Visibility = Visibility.Hidden;

BirthdayLabel.Visibility = Visibility.Hidden;

ChaireLabel.Visibility = Visibility.Hidden;

PositionLabel.Visibility = Visibility.Hidden;

YearStartLabel.Visibility = Visibility.Hidden;

SurnameText.Visibility = Visibility.Hidden;

NameText.Visibility = Visibility.Hidden;

FatherNameText.Visibility = Visibility.Hidden;

BirthdayText.Visibility = Visibility.Hidden;

ChaireText.Visibility = Visibility.Hidden;

PositionText.Visibility = Visibility.Hidden;

FacultiesLabel.Visibility = Visibility.Hidden;

YearStartText.Visibility = Visibility.Hidden;

KodLabel.Visibility = Visibility.Hidden;

KodText.Visibility = Visibility.Hidden;

OwnMarkLabel.Visibility = Visibility.Hidden;

OwnMarkText.Visibility = Visibility.Hidden;

GroupNameLabel.Visibility = Visibility.Hidden;

GroupNameText.Visibility = Visibility.Hidden;

HelpText.Visibility = Visibility.Hidden;

GroupNameLabel.Visibility = Visibility.Hidden;

GroupNameText.Visibility = Visibility.Hidden;

CourseLabel.Visibility = Visibility.Hidden;

CourseText.Visibility = Visibility.Hidden;

FacultyText.Visibility = Visibility.Hidden;

FacultiesLabel.Visibility = Visibility.Hidden;

resulting1.Content = "";

resulting2.Content = "";

resulting3.Content = "";

resulting4.Content = "";

resulting5.Content = "";

resulting6.Content = "";

resulting7.Content = "";

resulting8.Content = "";

resulting9.Content = "";

resulting10.Content = "";

resulting11.Content = "";

resulting12.Content = "";

SurnameText.Text = "";

NameText.Text = "";

FatherNameText.Text = "";

BirthdayText.Text = "";

ChaireText.Text = "";

PositionText.Text = "";

YearStartText.Text = "";

OwnMarkText.Text = "";

GroupNameText.Text = "";

CourseText.Text = "";

FacultyText.Text = "";

KodText.Text = "";

}

void VisibleTeacher()

{

SurnameLabel.Visibility = Visibility.Visible;

NameLabel.Visibility = Visibility.Visible;

FatherNameLabel.Visibility = Visibility.Visible;

BirthdayLabel.Visibility = Visibility.Visible;

ChaireLabel.Visibility = Visibility.Visible;

PositionLabel.Visibility = Visibility.Visible;

FacultiesLabel.Visibility = Visibility.Visible;

YearStartLabel.Visibility = Visibility.Visible;

SurnameText.Visibility = Visibility.Visible;

NameText.Visibility = Visibility.Visible;

FatherNameText.Visibility = Visibility.Visible;

BirthdayText.Visibility = Visibility.Visible;

ChaireText.Visibility = Visibility.Visible;

PositionText.Visibility = Visibility.Visible;

YearStartText.Visibility = Visibility.Visible;

KodLabel.Visibility = Visibility.Visible;

KodText.Visibility = Visibility.Visible;

OwnMarkLabel.Visibility = Visibility.Hidden;

OwnMarkText.Visibility = Visibility.Hidden;

GroupNameLabel.Visibility = Visibility.Hidden;

GroupNameText.Visibility = Visibility.Hidden;

Menu3.Visibility = Visibility.Hidden;

HelpText.Visibility = Visibility.Hidden;

FacultyText.Visibility = Visibility.Visible;

FacultiesLabel.Visibility = Visibility.Visible;

}

void VisibleStudent()

{

SurnameLabel.Visibility = Visibility.Visible;

NameLabel.Visibility = Visibility.Visible;

FatherNameLabel.Visibility = Visibility.Visible;

BirthdayLabel.Visibility = Visibility.Visible;

SurnameText.Visibility = Visibility.Visible;

NameText.Visibility = Visibility.Visible;

FatherNameText.Visibility = Visibility.Visible;

BirthdayText.Visibility = Visibility.Visible;

KodLabel.Visibility = Visibility.Visible;

KodText.Visibility = Visibility.Visible;

OwnMarkLabel.Visibility = Visibility.Visible;

OwnMarkText.Visibility = Visibility.Visible;

GroupNameLabel.Visibility = Visibility.Visible;

GroupNameText.Visibility = Visibility.Visible;

YearStartLabel.Visibility = Visibility.Visible;

YearStartLabel.Content = "Рік вступу";

YearStartText.Visibility = Visibility.Visible;

CourseLabel.Visibility = Visibility.Visible;

CourseText.Visibility = Visibility.Visible;

FacultyText.Visibility = Visibility.Visible;

FacultiesLabel.Visibility = Visibility.Visible;

}

public AdminOpportunity()

{

InitializeComponent();

ChangeLabel.Visibility = Visibility.Hidden;

AddLabel.Visibility = Visibility.Hidden;

HiddenAll();

}

private void Button\_Click(object sender, RoutedEventArgs e)// додати

{

AddLabel.Visibility = Visibility.Visible;

HiddenAll();

ChangeLabel.Visibility = Visibility.Hidden;

DeleteLabel1.Visibility = Visibility.Hidden;

DeleteLabel2.Visibility = Visibility.Hidden;

DeleteText.Visibility = Visibility.Hidden;

DeleteLabel4.Visibility = Visibility.Hidden;

Menu1.Visibility = Visibility.Hidden;

Menu1.SelectedIndex = -1;

Menu.SelectedIndex = -1;

Menu.Visibility = Visibility.Visible;

AddButton.Visibility = Visibility.Visible;

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)//видалити

{

HiddenAll();

ChangeLabel.Visibility = Visibility.Hidden;

AddLabel.Visibility = Visibility.Hidden;

DeleteLabel1.Visibility = Visibility.Visible;

DeleteLabel2.Visibility = Visibility.Visible;

DeleteText.Visibility = Visibility.Visible;

DeleteLabel4.Visibility = Visibility.Visible;

DeleteButton.Visibility = Visibility.Visible;

}

private void Button\_Click\_2(object sender, RoutedEventArgs e)//змінити

{

HiddenAll();

ChangeLabel.Visibility = Visibility.Visible;

AddLabel.Visibility = Visibility.Hidden;

ChangeButton.Visibility = Visibility.Visible;

Menu1.Visibility = Visibility.Visible;

Menu1.SelectedIndex = -1;

}

private void Button\_Click\_3(object sender, RoutedEventArgs e)//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!переглянути

{

HiddenAll();

AddLabel.Visibility = Visibility.Hidden;

ChangeLabel.Visibility = Visibility.Hidden;

File.Copy(@"G:\KursovaFleshechna\WPFMain\bin\Debug\course\_project.txt", @"G:\KursovaFleshechna\course\_project.txt", true);

}

private void DeleteButton\_Click(object sender, RoutedEventArgs e)//остаточно видалити

{

HiddenAll();

ChangeLabel.Visibility = Visibility.Hidden;

AddLabel.Visibility = Visibility.Hidden;

DeleteLabel1.Visibility = Visibility.Visible;

DeleteLabel2.Visibility = Visibility.Visible;

DeleteText.Visibility = Visibility.Visible;

DeleteLabel4.Visibility = Visibility.Visible;

DeleteButton.Visibility = Visibility.Visible;

uint koD;

bool help = uint.TryParse(DeleteText.Text, out koD);

if (DeleteLECTURERFromFile(koD))

{ MessageBox.Show("Елемент типу ВИКЛАДАЧ було успішно видалено з бази даних!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information); }

else

{

if (DeleteSTUDENTFromFile(koD))

{ MessageBox.Show("Елемент типу СТУДЕНТ було успішно видалено з бази даних!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information); }

}

string path\_1 = @"G:\KursovaFleshechna\WPFMain\bin\Debug\course\_project1.txt";

string path\_ = @"G:\KursovaFleshechna\WPFMain\bin\Debug\course\_project.txt";

string[] readText={""};

if (File.Exists(path\_1))

{

StreamReader sr = new StreamReader(path\_1, System.Text.Encoding.GetEncoding(1251));

string[] lines={sr.ReadToEnd()};

// MessageBox.Show("Вдало відкрили, считали!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information);

if (File.Exists(path\_))

{

File.WriteAllLines(path\_, lines);

// MessageBox.Show("Вдало відкрили, перезаписали!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

private void AddButton\_Click(object sender, RoutedEventArgs e)//продовжити додати

{

int choose = Menu.SelectedIndex;

switch (choose)

{

case 0://vukladach

{

HiddenAll();

Menu.Visibility = Visibility.Visible;

AddFinishButton.Visibility = Visibility.Visible;

VisibleTeacher();

HelpForLecturer.Visibility = Visibility.Visible;

break;

}

case 1://student

{

HiddenAll();

Menu.Visibility = Visibility.Visible;

Menu3.Visibility = Visibility.Visible;

AddButton2.Visibility = Visibility.Visible;

HelpForStudent.Visibility = Visibility.Visible;

break;

}

default: break;

}

}

private void AddFinishButton\_Click(object sender, RoutedEventArgs e)//додати запис остаточно

{

AddButton.Visibility = Visibility.Hidden;

if (Menu.SelectedIndex == 0 | (Menu.SelectedIndex == 1 && Menu3.SelectedIndex == 1))//lecturer or student

{

resulting2.Visibility = Visibility.Visible;

resulting1.Visibility = Visibility.Visible;

resulting3.Visibility = Visibility.Visible;

resulting4.Visibility = Visibility.Visible;

resulting5.Visibility = Visibility.Visible;

resulting6.Visibility = Visibility.Visible;

resulting7.Visibility = Visibility.Visible;

Menu.Visibility = Visibility.Visible;

Regex reg = new Regex(@"[А-ЯЇІЄЙ]([а-яйїіє]){1,}");

Match m = reg.Match(SurnameText.Text); //Прізвище

if (m.Success)

{

resulting1.Foreground = Brushes.Green;

resulting1.Content = "OK";

}

else

{

resulting1.Foreground = Brushes.Red;

resulting1.Content = "Неправильно";

}

reg = new Regex(@"[А-ЯЇІЄЙ]([а-яйїіє]){1,}");

m = reg.Match(NameText.Text); //Ім’я

if (m.Success)

{

resulting2.Foreground = Brushes.Green;

resulting2.Content = "OK";

}

else

{

resulting2.Foreground = Brushes.Red;

resulting2.Content = "Неправильно";

}

reg = new Regex(@"[А-ЯЇІЄЙ]([а-яйїіє]){1,}");

m = reg.Match(FatherNameText.Text); //По батькові

if (m.Success)

{

resulting3.Foreground = Brushes.Green;

resulting3.Content = "OK";

}

else

{

resulting3.Foreground = Brushes.Red;

resulting3.Content = "Неправильно";

}

reg = new Regex(@"((((0[1-9])|(1\d)|(2[01234]))\.(01|02|03|04|05|06|07|08|09|10|11|12))|(2[56789]|30)\.(01|03|04|05|06|07|08|09|10|11|12)|(31\.(01|03|05|07|08|10|12)))\.((19[1-9]\d)|(200[012]))");

m = reg.Match(BirthdayText.Text); //Дата народження

if (m.Success)

{

resulting4.Foreground = Brushes.Green;

resulting4.Content = "OK";

}

else

{

resulting4.Foreground = Brushes.Red;

resulting4.Content = "Неправильно";

}

reg = new Regex(@"(196[2-9]|19[789]\d|200\d|201[0-7])");

m = reg.Match(YearStartText.Text); //Рік старт

if (m.Success)

{

resulting5.Foreground = Brushes.Green;

resulting5.Content = "OK";

}

else

{

resulting5.Foreground = Brushes.Red;

resulting5.Content = "Неправильно";

}

reg = new Regex(@"\d{10}");

m = reg.Match(KodText.Text); //Ідентифікаціний код

if (m.Success)

{

resulting6.Foreground = Brushes.Green;

resulting6.Content = "OK";

}

else

{

resulting6.Foreground = Brushes.Red;

resulting6.Content = "Неправильно";

}

reg = new Regex(@"(([А-ЯІЇЄЮЙЕ]{3,4}(.|)){1,5})");

m = reg.Match(FacultyText.Text); //Факультет

if (m.Success)

{

resulting7.Foreground = Brushes.Green;

resulting7.Content = "OK";

}

else

{

resulting7.Foreground = Brushes.Red;

resulting7.Content = "Неправильно";

}

}

if (Menu.SelectedIndex == 0) //lecturer

{

resulting11.Visibility = Visibility.Visible;

resulting12.Visibility = Visibility.Visible;

Regex reg1 = new Regex(@"КТМ|(КА(іАГ)|(тКІТ))|(КФ(тВМ)|(ВіС)|(ФіК))|КЗД|(КГ(М)|(іСН))|(КК(СУтА)|І)|КПЗС|(КР(ДАтТ)|(РКК))|(КМ(ОіА)|Е)|КМ|(КЕ(тП)|(БПУтА))|КЕ|КІМ|КОіА");

Match m1 = reg1.Match(ChaireText.Text); //Кафедра

if (m1.Success)

{

resulting11.Foreground = Brushes.Green;

resulting11.Content = "OK";

}

else

{

resulting11.Foreground = Brushes.Red;

resulting11.Content = "Неправильно";

}

reg1 = new Regex(@"[А-ЯЇІЄЙ]([а-я]){1,}");

m1 = reg1.Match(PositionText.Text); //Посада

if (m1.Success)

{

resulting12.Foreground = Brushes.Green;

resulting12.Content = "OK";

}

else

{

resulting12.Foreground = Brushes.Red;

resulting12.Content = "Неправильно";

}

//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

if (SurnameText.Text != null && NameText.Text != null && FatherNameText.Text != null && BirthdayText.Text != null &&

YearStartText.Text != null && KodText.Text != null && FacultyText.Text != null && ChaireText.Text != null && PositionText.Text != null

&&resulting1.Content == "OK"&&resulting2.Content == "OK"&&resulting3.Content == "OK"&&resulting4.Content == "OK"&&

resulting5.Content == "OK"&&resulting7.Content == "OK"&&resulting11.Content == "OK"&&resulting12.Content == "OK")

{

uint tmpKod;

if (uint.TryParse(KodText.Text, out tmpKod))

{

if (!ExistLECTURER\_InFile(tmpKod))

{

Add\_And\_Print\_PointLECTURERToDictionary(path\_,ChaireText.Text, PositionText.Text, NameText.Text, SurnameText.Text, FatherNameText.Text,

tmpKod, BirthdayText.Text, YearStartText.Text, FacultyText.Text);

MessageBox.Show("Елемент типу ВИКЛАДАЧ було успішно додано до бази даних!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

}

if (Menu.SelectedIndex == 1 && Menu3.SelectedIndex == 1)//student

{

resulting8.Visibility = Visibility.Visible;

resulting9.Visibility = Visibility.Visible;

resulting10.Visibility = Visibility.Visible;

Regex reg1 = new Regex(@"[1-5]");

Match m1 = reg1.Match(CourseText.Text); //Курс

if (m1.Success)

{

resulting8.Foreground = Brushes.Green;

resulting8.Content = "OK";

}

else

{

resulting8.Foreground = Brushes.Red;

resulting8.Content = "Неправильно";

}

reg1 = new Regex(@"[А-ЯІЇЄЙ]{2,6}\-\d{1,3}");

m1 = reg1.Match(GroupNameText.Text); //Назва групи

if (m1.Success)

{

resulting9.Foreground = Brushes.Green;

resulting9.Content = "OK";

}

else

{

resulting9.Foreground = Brushes.Red;

resulting9.Content = "Неправильно";

}

reg1 = new Regex(@"((\d|\d{2}).\d)|(100.0)");

m1 = reg1.Match(OwnMarkText.Text); //Середній бал

if (m1.Success)

{

resulting10.Foreground = Brushes.Green;

resulting10.Content = "OK";

}

else

{

resulting10.Foreground = Brushes.Red;

resulting10.Content = "Неправильно";

}

//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

if (SurnameText.Text != null && NameText.Text != null && FatherNameText.Text != null && BirthdayText.Text != null &&

YearStartText.Text != null && KodText.Text != null && FacultyText.Text != null && CourseText.Text != null &&

OwnMarkText.Text != null && GroupNameText.Text != null && resulting1.Content == "OK" && resulting2.Content == "OK"

&& resulting3.Content == "OK" && resulting4.Content == "OK" && resulting5.Content == "OK" && resulting7.Content == "OK"

&& resulting8.Content == "OK" && resulting9.Content == "OK" && resulting10.Content == "OK")

{

uint tmpKod;

int tmpcourse;

double tmpmark;

if (uint.TryParse(KodText.Text, out tmpKod))

{

if (int.TryParse(CourseText.Text, out tmpcourse))

{

if (double.TryParse(OwnMarkText.Text, out tmpmark))

{

if (!ExistSTUDENT\_InFile(tmpKod))

{

Add\_And\_Print\_PointSTUDENTToDictionary(path\_,GroupNameText.Text, tmpcourse, tmpmark, NameText.Text, SurnameText.Text, FatherNameText.Text,

tmpKod, BirthdayText.Text, YearStartText.Text, FacultyText.Text);

MessageBox.Show("Елемент типу СТУДЕНТ було успішно додано до бази даних!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

}

}

}

if (Menu.SelectedIndex == 1 && Menu3.SelectedIndex == 0)// group of students

{

HiddenAll();

Menu.Visibility = Visibility.Visible;

Menu3.Visibility = Visibility.Visible;

AddFinishButton.Visibility = Visibility.Visible;

HelpText.Visibility = Visibility.Visible;

GroupNameLabel.Visibility = Visibility.Visible;

GroupNameText.Visibility = Visibility.Visible;

FacultiesLabel.Visibility = Visibility.Visible;

FacultyText.Visibility = Visibility.Visible;

CourseLabel.Visibility = Visibility.Visible;

CourseText.Visibility = Visibility.Visible;

AddButton2.Visibility = Visibility.Visible;

int tmpcourse=0;

string tmpgroup="";

string tmpfaculty="";

{

{

VisibleStudent();

resulting8.Visibility = Visibility.Visible;

resulting9.Visibility = Visibility.Visible;

resulting10.Visibility = Visibility.Visible;

Regex reg1 = new Regex(@"[1-5]");

Match m1 = reg1.Match(CourseText.Text); //Курс

if (m1.Success)

{

resulting8.Foreground = Brushes.Green;

resulting8.Content = "OK";

bool two = int.TryParse(CourseText.Text,out tmpcourse);

}

else

{

resulting8.Foreground = Brushes.Red;

resulting8.Content = "Неправильно";

}

reg1 = new Regex(@"[А-ЯІЇЄЙ]{2,6}\-\d{1,3}");

m1 = reg1.Match(GroupNameText.Text); //Назва групи

if (m1.Success)

{

resulting9.Foreground = Brushes.Green;

resulting9.Content = "OK";

tmpgroup = GroupNameText.Text;

}

else

{

resulting9.Foreground = Brushes.Red;

resulting9.Content = "Неправильно";

}

reg1 = new Regex(@"(Ф((ІМ)|(ІКТ))|Ф((УМ)|(ОФ)))|(ГЕФ)");

m1 = reg1.Match(FacultyText.Text); //Факультет

if (m1.Success)

{

resulting7.Foreground = Brushes.Green;

resulting7.Content = "OK";

tmpfaculty = FacultyText.Text;

}

else

{

resulting7.Foreground = Brushes.Red;

resulting7.Content = "Неправильно";

}

}

int identification\_for\_group = 0;

if (identification\_for\_group == 0)

{

resulting8.Visibility = Visibility.Hidden;

resulting9.Visibility = Visibility.Hidden;

resulting10.Visibility = Visibility.Hidden;

GroupNameLabel.Visibility = Visibility.Hidden;

GroupNameText.Visibility = Visibility.Hidden;

FacultiesLabel.Visibility = Visibility.Hidden;

FacultyText.Visibility = Visibility.Hidden;

CourseLabel.Visibility = Visibility.Hidden;

CourseText.Visibility = Visibility.Hidden;

resulting2.Visibility = Visibility.Visible; //если в групе много персон

resulting1.Visibility = Visibility.Visible;

resulting3.Visibility = Visibility.Visible;

resulting4.Visibility = Visibility.Visible;

resulting5.Visibility = Visibility.Visible;

resulting6.Visibility = Visibility.Visible;

resulting7.Visibility = Visibility.Visible;

Menu.Visibility = Visibility.Visible;

identification\_for\_group++;

}

resulting7.Content = "";

Regex reg = new Regex(@"[А-ЯЇІЄЙ]([а-я]){1,}");

Match m = reg.Match(SurnameText.Text); //Прізвище

if (m.Success)

{

resulting1.Foreground = Brushes.Green;

resulting1.Content = "OK";

}

else

{

resulting1.Foreground = Brushes.Red;

resulting1.Content = "Неправильно";

}

reg = new Regex(@"[А-ЯЇІЄЙ]([а-я]){1,}");

m = reg.Match(NameText.Text); //Ім’я

if (m.Success)

{

resulting2.Foreground = Brushes.Green;

resulting2.Content = "OK";

}

else

{

resulting2.Foreground = Brushes.Red;

resulting2.Content = "Неправильно";

}

reg = new Regex(@"([А-ЯЇІЄЙ](?:[а-яіїйє]){2,}((?:ович)|(?:вна)))");

m = reg.Match(FatherNameText.Text); //По батькові

if (m.Success)

{

resulting3.Foreground = Brushes.Green;

resulting3.Content = "OK";

}

else

{

resulting3.Foreground = Brushes.Red;

resulting3.Content = "Неправильно";

}

reg = new Regex(@"((((0[1-9])|(1\d)|(2[01234]))\.(01|02|03|04|05|06|07|08|09|10|11|12))|(2[56789]|30)\.(01|03|04|05|06|07|08|09|10|11|12)|(31\.(01|03|05|07|08|10|12)))\.((19[1-9]\d)|(200[012]))");

m = reg.Match(BirthdayText.Text); //Дата народження

if (m.Success)

{

resulting4.Foreground = Brushes.Green;

resulting4.Content = "OK";

}

else

{

resulting4.Foreground = Brushes.Red;

resulting4.Content = "Неправильно";

}

reg = new Regex(@"(196[2-9]|19[789]\d|200\d|201[0-7])");

m = reg.Match(YearStartText.Text); //Рік старт

if (m.Success)

{

resulting5.Foreground = Brushes.Green;

resulting5.Content = "OK";

}

else

{

resulting5.Foreground = Brushes.Red;

resulting5.Content = "Неправильно";

}

reg = new Regex(@"\d{10}");

m = reg.Match(KodText.Text); //Ідентифікаціний код

if (m.Success)

{

resulting6.Foreground = Brushes.Green;

resulting6.Content = "OK";

}

else

{

resulting6.Foreground = Brushes.Red;

resulting6.Content = "Неправильно";

}

reg = new Regex(@"((\d|\d{2}).\d)|(100.0)");

m = reg.Match(OwnMarkText.Text); //Середній бал

if (m.Success)

{

resulting10.Foreground = Brushes.Green;

resulting10.Content = "OK";

}

else

{

resulting10.Foreground = Brushes.Red;

resulting10.Content = "Неправильно";

}

//!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

if (SurnameText.Text != null && NameText.Text != null && FatherNameText.Text != null && BirthdayText.Text != null &&

YearStartText.Text != null && KodText.Text != null && tmpfaculty!= null && tmpgroup != null &&

OwnMarkText.Text != null && tmpcourse != null && resulting1.Content == "OK" && resulting2.Content == "OK"

&& resulting3.Content == "OK" && resulting4.Content == "OK" && resulting5.Content == "OK" && resulting7.Content == "OK"

&& resulting8.Content == "OK" && resulting9.Content == "OK" && resulting10.Content == "OK")

{

uint tmpKod;

double tmpmark;

if (uint.TryParse(KodText.Text, out tmpKod))

{

if (int.TryParse(CourseText.Text, out tmpcourse))

{

if (double.TryParse(OwnMarkText.Text, out tmpmark))

{

if (!ExistSTUDENT\_InFile(tmpKod))

{

Add\_And\_Print\_PointSTUDENTToDictionary(path\_,tmpgroup, tmpcourse, tmpmark, NameText.Text, SurnameText.Text, FatherNameText.Text,

tmpKod, BirthdayText.Text, YearStartText.Text, tmpfaculty);

MessageBox.Show("Елемент типу СТУДЕНТ було успішно додано до бази даних!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

}

}

}

}

}

private void ChangeButton\_Click(object sender, RoutedEventArgs e)//продовжити змінити

{

int choose3 = Menu1.SelectedIndex;

HelpText.Visibility = Visibility.Hidden;

switch (choose3)

{

case 0://vukladach

{

VisibleTeacher();

HelpForLecturer.Visibility = Visibility.Visible;

break;

}

case 1://1 student

{

VisibleStudent();

HelpForStudent.Visibility = Visibility.Visible;

break;

}

default: break;

}

ChangeFinishButton.Visibility = Visibility.Visible;

}

private void ChangeFinishButton\_Click(object sender, RoutedEventArgs e)//змінити остаточно

{

resulting2.Visibility = Visibility.Visible;

resulting1.Visibility = Visibility.Visible;

resulting3.Visibility = Visibility.Visible;

resulting4.Visibility = Visibility.Visible;

resulting5.Visibility = Visibility.Visible;

resulting6.Visibility = Visibility.Visible;

resulting7.Visibility = Visibility.Visible;

int helping = 0;

if (Menu1.SelectedIndex == 0 | Menu1.SelectedIndex == 1)//lecturer or student

{

ChangeButton.Visibility = Visibility.Hidden;

Regex reg = new Regex(@"[А-ЯЇІЄЙ]([а-яіїяюєй]){1,}");

Match m = reg.Match(SurnameText.Text); //Прізвище

if (m.Success)

{

resulting1.Foreground = Brushes.Green;

resulting1.Content = "OK";

helping++;

}

else

{

resulting1.Foreground = Brushes.Red;

resulting1.Content = "Неправильно";

}

reg = new Regex(@"[А-ЯЇІЄЙ]([а-яіїяюєй]){1,}");

m = reg.Match(NameText.Text); //Ім’я

if (m.Success)

{

resulting2.Foreground = Brushes.Green;

resulting2.Content = "OK";

helping++;

}

else

{

resulting2.Foreground = Brushes.Red;

resulting2.Content = "Неправильно";

}

reg = new Regex(@"[А-ЯЇІЄЙ]([а-яіїяюєй]){1,}");

m = reg.Match(FatherNameText.Text); //По батькові

if (m.Success)

{

resulting3.Foreground = Brushes.Green;

resulting3.Content = "OK";

helping++;

}

else

{

resulting3.Foreground = Brushes.Red;

resulting3.Content = "Неправильно";

}

reg = new Regex(@"((((0[1-9])|(1\d)|(2[01234]))\.(01|02|03|04|05|06|07|08|09|10|11|12))|(2[56789]|30)\.(01|03|04|05|06|07|08|09|10|11|12)|(31\.(01|03|05|07|08|10|12)))\.((19[1-9]\d)|(200[012]))");

m = reg.Match(BirthdayText.Text); //Дата народження

if (m.Success)

{

resulting4.Foreground = Brushes.Green;

resulting4.Content = "OK";

helping++;

}

else

{

resulting4.Foreground = Brushes.Red;

resulting4.Content = "Неправильно";

}

reg = new Regex(@"(196[2-9]|19[789]\d|200\d|201[0-7])");

m = reg.Match(YearStartText.Text); //Рік старт

if (m.Success)

{

resulting5.Foreground = Brushes.Green;

resulting5.Content = "OK";

helping++;

}

else

{

resulting5.Foreground = Brushes.Red;

resulting5.Content = "Неправильно";

}

reg = new Regex(@"\d{10}");

m = reg.Match(KodText.Text); //Ідентифікаціний код

if (m.Success)

{

resulting6.Foreground = Brushes.Green;

resulting6.Content = "OK";

helping++;

}

else

{

resulting6.Foreground = Brushes.Red;

resulting6.Content = "Неправильно";

}

reg = new Regex(@"(([А-ЯІЇЄЮЙЕ]{3,4}(.|)){1,5})");

m = reg.Match(FacultyText.Text); //Факультет

if (m.Success)

{

resulting7.Foreground = Brushes.Green;

resulting7.Content = "OK";

helping++;

}

else

{

resulting7.Foreground = Brushes.Red;

resulting7.Content = "Неправильно";

}

}

if (Menu1.SelectedIndex == 1)//student

{

resulting8.Visibility = Visibility.Visible;

resulting9.Visibility = Visibility.Visible;

resulting10.Visibility = Visibility.Visible;

Regex reg1 = new Regex(@"[1-5]");

Match m1 = reg1.Match(CourseText.Text); //Курс

if (m1.Success)

{

resulting8.Foreground = Brushes.Green;

resulting8.Content = "OK";

helping++;

}

else

{

resulting8.Foreground = Brushes.Red;

resulting8.Content = "Неправильно";

}

reg1 = new Regex(@"[А-ЯІЇЄЙ]{2,6}\-\d{1,3}");

m1 = reg1.Match(GroupNameText.Text); //Назва групи

if (m1.Success)

{

resulting9.Foreground = Brushes.Green;

resulting9.Content = "OK";

helping++;

}

else

{

resulting9.Foreground = Brushes.Red;

resulting9.Content = "Неправильно";

}

reg1 = new Regex(@"((\d|\d{2}).\d)|(100.0)");

m1 = reg1.Match(OwnMarkText.Text); //Середній бал

if (m1.Success)

{

resulting10.Foreground = Brushes.Green;

resulting10.Content = "OK";

helping++;

}

else

{

resulting10.Foreground = Brushes.Red;

resulting10.Content = "Неправильно";

}

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

uint KOD;

string path\_2 = @"course\_project1.txt";

if (uint.TryParse(KodText.Text, out KOD))

{

if (DeleteSTUDENTFromFile(KOD))

{ MessageBox.Show("Елемент типу СТУДЕНТ було успішно видалено з бази даних!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information); }

}

if (SurnameText.Text != null && NameText.Text != null && FatherNameText.Text != null && BirthdayText.Text != null &&

YearStartText.Text != null && KodText.Text != null && FacultyText.Text != null && GroupNameText.Text != null &&

OwnMarkText.Text != null && CourseText.Text != null && helping == 10)

{

double tmpmark;

int tmpcourse = -1;

if (uint.TryParse(KodText.Text, out KOD))

{

if (int.TryParse(CourseText.Text, out tmpcourse))

{

if (double.TryParse(OwnMarkText.Text, out tmpmark))

{

Add\_And\_Print\_PointSTUDENTToDictionary(path\_2, GroupNameText.Text, tmpcourse, tmpmark, NameText.Text, SurnameText.Text, FatherNameText.Text,

KOD, BirthdayText.Text, YearStartText.Text, FacultyText.Text);

MessageBox.Show("Елемент типу СТУДЕНТ було успішно змінено в базі даних!", "Повідомлення!", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

}

string path\_1 = @"G:\KursovaFleshechna\WPFMain\bin\Debug\course\_project1.txt";

string path\_ = @"G:\KursovaFleshechna\WPFMain\bin\Debug\course\_project.txt";

if (File.Exists(path\_1))

{

StreamReader sr = new StreamReader(path\_1, System.Text.Encoding.GetEncoding(1251));

string[] lines = { sr.ReadToEnd() };

MessageBox.Show("Вдало відкрили, считали!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information);

if (File.Exists(path\_))

{

File.WriteAllLines(path\_, lines);

MessageBox.Show("Вдало відкрили, перезаписали!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

if (Menu1.SelectedIndex == 0)//lecturer

{

resulting11.Visibility = Visibility.Visible;

resulting12.Visibility = Visibility.Visible;

Regex reg1 = new Regex(@"КТМ|(КА(іАГ)|(тКІТ))|(КФ(тВМ)|(ВіС)|(ФіК))|КЗД|(КГ(М)|(іСН))|(КК(СУтА)|І)|КПЗС|(КР(ДАтТ)|(РКК))|(КМ(ОіА)|()|Е)|(КЕ(тП)|()|(БПУтА))|КІМ|КОіА");

Match m1 = reg1.Match(ChaireText.Text); //Кафедра

if (m1.Success)

{

resulting11.Foreground = Brushes.Green;

resulting11.Content = "OK";

helping++;

}

else

{

resulting11.Foreground = Brushes.Red;

resulting11.Content = "Неправильно";

}

reg1 = new Regex(@"[А-ЯЇІЄЙ]([а-я]){1,}");

m1 = reg1.Match(PositionText.Text); //Посада

if (m1.Success)

{

resulting12.Foreground = Brushes.Green;

resulting12.Content = "OK";

helping++;

}

else

{

resulting12.Foreground = Brushes.Red;

resulting12.Content = "Неправильно";

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

uint KOD;

string path\_3 = @"course\_project1.txt";

if (uint.TryParse(KodText.Text, out KOD))

{

if (DeleteLECTURERFromFile(KOD))

{ MessageBox.Show("Елемент типу ВИКЛАДАЧ було успішно видалено з бази даних!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information); }

}

if (SurnameText.Text != null && NameText.Text != null && FatherNameText.Text != null && BirthdayText.Text != null &&

YearStartText.Text != null && KodText.Text != null && FacultyText.Text != null && ChaireText.Text != null && PositionText.Text != null

&& helping == 9)

{

uint tmpKod;

if (uint.TryParse(KodText.Text, out tmpKod))

{

Add\_And\_Print\_PointLECTURERToDictionary(path\_3,ChaireText.Text, PositionText.Text, NameText.Text, SurnameText.Text, FatherNameText.Text,

tmpKod, BirthdayText.Text, YearStartText.Text, FacultyText.Text);

MessageBox.Show("Елемент типу ВИКЛАДАЧ було успішно змінено в базі даних!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

string path\_1 = @"G:\KursovaFleshechna\WPFMain\bin\Debug\course\_project1.txt";

string path\_ = @"G:\KursovaFleshechna\WPFMain\bin\Debug\course\_project.txt";

string[] readText = { "" };

if (File.Exists(path\_1))

{

StreamReader sr = new StreamReader(path\_1, System.Text.Encoding.GetEncoding(1251));

string[] lines = { sr.ReadToEnd() };

MessageBox.Show("Вдало відкрили, считали!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information);

if (File.Exists(path\_))

{

File.WriteAllLines(path\_, lines);

MessageBox.Show("Вдало відкрили, перезаписали!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

}

}

private void AddButton2\_Click(object sender, RoutedEventArgs e)//для студента продовжуємо додавати//тут створюэться група

{

AddButton.Visibility = Visibility.Hidden;

Menu3.Visibility = Visibility.Visible;

int choose2 = Menu3.SelectedIndex;

int helping = 0;

switch (choose2)

{

case 0://group

{

AddFinishButton.Visibility = Visibility.Visible;

HelpText.Visibility = Visibility.Visible;

VisibleStudent();

resulting8.Visibility = Visibility.Visible;

resulting9.Visibility = Visibility.Visible;

resulting10.Visibility = Visibility.Visible;

GroupNameLabel.Visibility = Visibility.Visible;

GroupNameText.Visibility = Visibility.Visible;

FacultiesLabel.Visibility = Visibility.Visible;

FacultyText.Visibility = Visibility.Visible;

CourseLabel.Visibility = Visibility.Visible;

CourseText.Visibility = Visibility.Visible;

AddButton2.Visibility = Visibility.Visible;

int tmpcourse = 0;

string tmpgroup="";

string tmpfaculty="";

{

{

if ((GroupNameText.Text != null) && (FacultyText.Text != null) && (CourseText.Text != null))

{ resulting8.Visibility = Visibility.Visible;

resulting9.Visibility = Visibility.Visible;

resulting10.Visibility = Visibility.Visible;

Regex reg1 = new Regex(@"[1-5]");

Match m1 = reg1.Match(CourseText.Text); //Курс

if (m1.Success)

{

resulting8.Foreground = Brushes.Green;

resulting8.Content = "OK";

bool two = int.TryParse(CourseText.Text, out tmpcourse);

helping++;

}

else

{

resulting8.Foreground = Brushes.Red;

resulting8.Content = "Неправильно";

}

reg1 = new Regex(@"[А-ЯІЇЄЙ]{2,6}\-\d{1,3}");

m1 = reg1.Match(GroupNameText.Text); //Назва групи

if (m1.Success)

{

resulting9.Foreground = Brushes.Green;

resulting9.Content = "OK";

tmpgroup = GroupNameText.Text;

helping++;

}

else

{

resulting9.Foreground = Brushes.Red;

resulting9.Content = "Неправильно";

}

reg1 = new Regex(@"(Ф((ІМ)|(ІКТ))|Ф((УМ)|(ОФ)))|(ГЕФ)");

m1 = reg1.Match(FacultyText.Text); //Факультет

if (m1.Success)

{

resulting7.Foreground = Brushes.Green;

resulting7.Content = "OK";

tmpfaculty = FacultyText.Text;

helping++;

}

else

{

resulting7.Foreground = Brushes.Red;

resulting7.Content = "Неправильно";

}

}

}

VisibleStudent();

resulting1.Content = "";

resulting2.Content = "";

resulting3.Content = "";

resulting4.Content = "";

resulting5.Content = "";

resulting6.Content = "";

resulting7.Content = "";

resulting8.Content = "";

resulting9.Content = "";

resulting11.Content = "";

resulting12.Content = "";

resulting8.Visibility = Visibility.Hidden;

resulting9.Visibility = Visibility.Hidden;

resulting10.Visibility = Visibility.Hidden;

GroupNameLabel.Visibility = Visibility.Hidden;

GroupNameText.Visibility = Visibility.Hidden;

FacultiesLabel.Visibility = Visibility.Hidden;

FacultyText.Visibility = Visibility.Hidden;

CourseLabel.Visibility = Visibility.Hidden;

CourseText.Visibility = Visibility.Hidden;

resulting2.Visibility = Visibility.Visible; //если в групе много персон

resulting1.Visibility = Visibility.Visible;

resulting3.Visibility = Visibility.Visible;

resulting4.Visibility = Visibility.Visible;

resulting5.Visibility = Visibility.Visible;

resulting6.Visibility = Visibility.Visible;

resulting7.Visibility = Visibility.Visible;

Menu.Visibility = Visibility.Visible;

Regex reg = new Regex(@"[А-ЯЇІЄЙ]([а-яїієьй]){1,}");

Match m = reg.Match(SurnameText.Text); //Прізвище

if (m.Success)

{

resulting1.Foreground = Brushes.Green;

resulting1.Content = "OK";

}

else

{

resulting1.Foreground = Brushes.Red;

resulting1.Content = "Неправильно";

}

reg = new Regex(@"[А-ЯЇІЄЙ]([а-яїієьй]){1,}");

m = reg.Match(NameText.Text); //Ім’я

if (m.Success)

{

resulting2.Foreground = Brushes.Green;

resulting2.Content = "OK";

}

else

{

resulting2.Foreground = Brushes.Red;

resulting2.Content = "Неправильно";

}

reg = new Regex(@"[А-ЯЇІЄЙ]([а-яїієьй]){1,}");

m = reg.Match(FatherNameText.Text); //По батькові

if (m.Success)

{

resulting3.Foreground = Brushes.Green;

resulting3.Content = "OK";

}

else

{

resulting3.Foreground = Brushes.Red;

resulting3.Content = "Неправильно";

}

reg = new Regex(@"((((0[1-9])|(1\d)|(2[01234]))\.(01|02|03|04|05|06|07|08|09|10|11|12))|(2[56789]|30)\.(01|03|04|05|06|07|08|09|10|11|12)|(31\.(01|03|05|07|08|10|12)))\.((19[1-9]\d)|(200[012]))");

m = reg.Match(BirthdayText.Text); //Дата народження

if (m.Success)

{

resulting4.Foreground = Brushes.Green;

resulting4.Content = "OK";

}

else

{

resulting4.Foreground = Brushes.Red;

resulting4.Content = "Неправильно";

}

reg = new Regex(@"(196[2-9]|19[789]\d|200\d|201[0-7])");

m = reg.Match(YearStartText.Text); //Рік старт

if (m.Success)

{

resulting5.Foreground = Brushes.Green;

resulting5.Content = "OK";

}

else

{

resulting5.Foreground = Brushes.Red;

resulting5.Content = "Неправильно";

}

reg = new Regex(@"\d{10}");

m = reg.Match(KodText.Text); //Ідентифікаціний код

if (m.Success)

{

resulting6.Foreground = Brushes.Green;

resulting6.Content = "OK";

}

else

{

resulting6.Foreground = Brushes.Red;

resulting6.Content = "Неправильно";

}

reg = new Regex(@"((\d|\d{2}).\d)|(100.0)");

m = reg.Match(OwnMarkText.Text); //Середній бал

if (m.Success)

{

resulting10.Foreground = Brushes.Green;

resulting10.Content = "OK";

}

else

{

resulting10.Foreground = Brushes.Red;

resulting10.Content = "Неправильно";

}

//!!!!!!!!!!!!!!!!!!!\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if (SurnameText.Text != null && NameText.Text != null && FatherNameText.Text != null && BirthdayText.Text != null &&

YearStartText.Text != null && KodText.Text != null && tmpfaculty != null && tmpgroup != null &&

OwnMarkText.Text != null && tmpcourse != null )//

{

uint tmpKod;

double tmpmark;

bool one = uint.TryParse(KodText.Text, out tmpKod);

bool two = double.TryParse(OwnMarkText.Text, out tmpmark);

if (!ExistSTUDENT\_InFile(tmpKod))

{

Add\_And\_Print\_PointSTUDENTToDictionary(path\_,tmpgroup, tmpcourse, tmpmark, NameText.Text, SurnameText.Text, FatherNameText.Text,

tmpKod, BirthdayText.Text, YearStartText.Text, tmpfaculty);

MessageBox.Show("Елемент типу СТУДЕНТ було успішно додано до бази даних!", "Attention", MessageBoxButton.OK, MessageBoxImage.Information);

}

} break;

}

}

case 1:

{

AddFinishButton.Visibility = Visibility.Visible;

HelpText.Visibility = Visibility.Hidden;

VisibleStudent();//1 student

AddButton2.Visibility = Visibility.Hidden;

break;

}

default: break;

}

}

}

}