Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

Лабораторна робота № 2

із дисципліни «Технології розробки програмного забезпечення»

Тема: «Основи проектування.»

Виконав
Студенти групи ІА-31:
Корнійчук М.Р

Перевірив:
Мягкий М.Ю

Київ 2025

Тема: Основи проектування.

Мета: Обрати зручну систему побудови UML-діаграм та навчитися будувати діаграми варіантів використання для системи що проєктується, розробляти сценарії варіантів використання та будувати діаграми класів предметної області.

Завдання:

• Ознайомитись з короткими теоретичними відомостями.

• Проаналізувати тему та спроєктувати діаграму варіантів використання відповідно до обраної теми лабораторного циклу.

• Спроєктувати діаграму класів предметної області.

• Вибрати 3 варіанти використання та написати за ними сценарії використання.

• На основі спроєктованої діаграми класів предметної області розробити основні класи та структуру бази даних системи. Класи даних повинні реалізуватишаблон Repository для взаємодії з базою даних.

• Нарисувати діаграму класів для реалізованої частини системи.

• Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму варіантів використання відповідно, діаграму класів системи, вихідні коди класів системи, а також зображення структури бази даних.

Хід роботи:

Тема проекту: Text Editor MK (repository, observer, strategy, factory, singleton)

Функціональність: Відображення текстових документів з підтримкою різних кодувань. Валідація та збереження файлів. Створення зручного інтерфейсу для читання та редагування текстових документів з метаданими (ім'я файлу, шлях, дата створення, кодування тощо). Автоматичне збереження при редагуванні, історія нещодавніх файлів. Перевірка статусу збереження документа. Налаштування редактора (шрифт, тема, перенос рядків).

Опис:

Актор - User (Користувач) - взаємодіє з системою та має доступ до таких функцій:

- Створити новий документ
- Відкрити існуючий файл
- Редагувати текст документа
- Зберегти документ
- Зберегти документ як...
- Вибрати кодування тексту
- Переглянути нещодавні файли
- Налаштувати редактор (шрифт, тема)
- Закрити документ
- Вийти з програми

Варіанти використання:

1. Управління документами

- Створення документа - створення нового порожнього текстового документа в системі
- Відкриття документа - завантаження існуючого текстового файлу з диска
- Збереження документа – запис поточного документа на диск з обраним кодуванням
- Закриття документа - завершення роботи з поточним документом

2. Робота з текстовим контентом

- Редагування тексту - внесення змін у текстовий контент документа
- Встановлення контенту - заміна всього тексту документа новим контентом
- Відстеження змін - автоматичне оновлення часу модифікації та статусу збереження
- Валідація кодування - перевірка та автоматичне визначення кодування тексту

3. Управління налаштуваннями

- Налаштування шрифту - зміна сімейства та розміру шрифту редактора
- Вибір теми - перемикання між світлою та темною темами оформлення
- Налаштування перенесення - увімкнення/вимкнення автоматичного перенесення рядків
- Відображення номерів рядків - показ/приховування нумерації рядків тексту

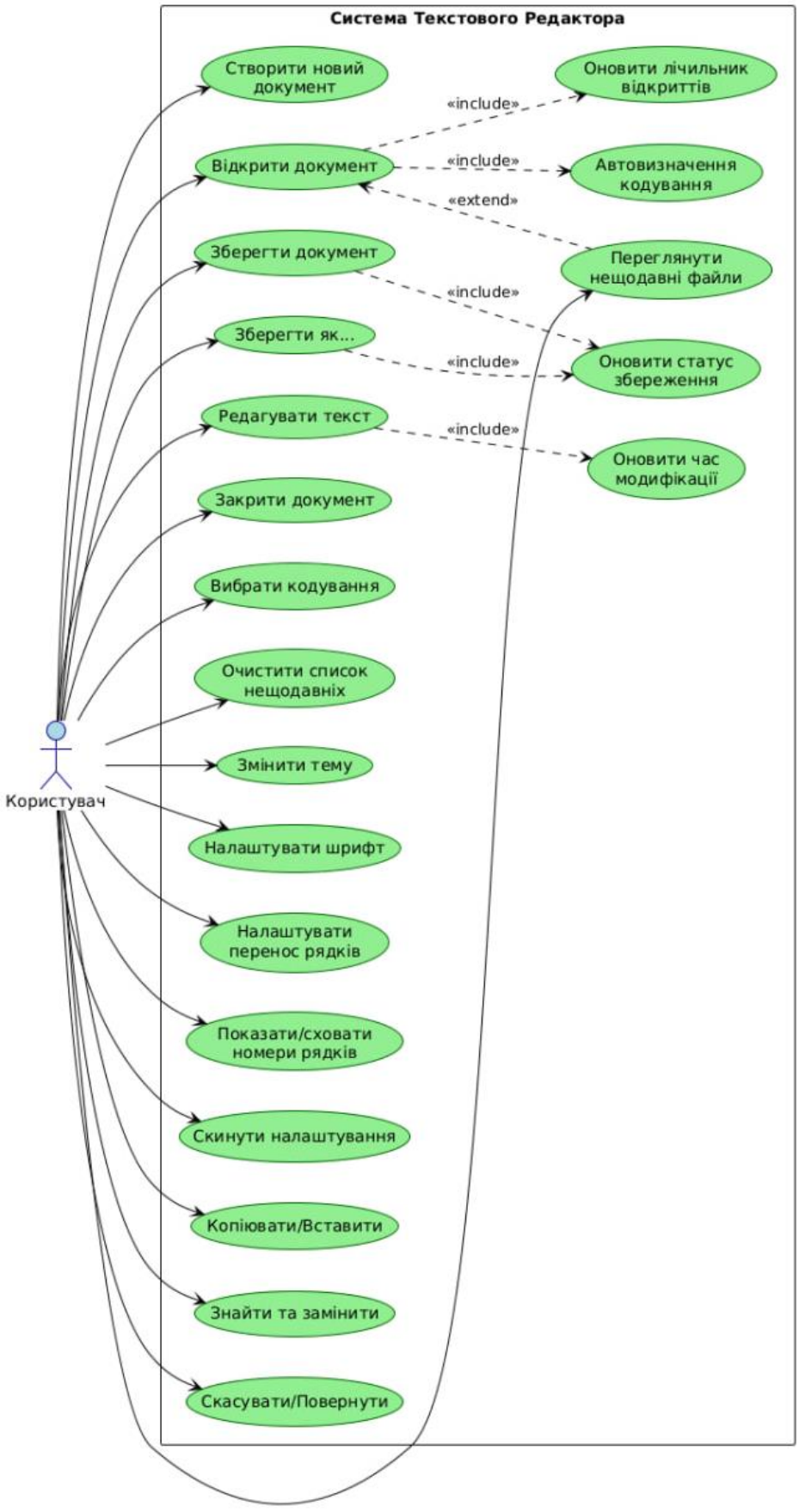
4. Історія та моніторинг

- Ведення списку нещодавніх - система зберігає історію відкритих файлів з часом доступу
- Підрахунок відкриттів - відстеження кількості разів відкриття кожного файлу
- Оновлення метаданих - автоматичне оновлення інформації про документи

5. Робота з кодуваннями

- Автоматичне визначення - система самостійно розпізнає кодування при відкритті файлів
- Вибір кодування - користувач може вручну обрати потрібне кодування для збереження
- Підтримка стандартів - робота з UTF-8, UTF-16 та іншими кодуваннями

Діаграма варіантів використання:



Клас Document моделює текстовий документ у системі TextEditor. Він відповідає за зберігання та управління текстовим контентом.

Атрибути:

- Id: int - унікальний ідентифікатор документа
- FileName: string - назва файлу документа
- FilePath: string - повний шлях до файлу в системі
- Content: string - текстовий вміст документа
- EncodingId: int - ідентифікатор кодування тексту
- CreatedAt: DateTime - дата та час створення документа
- ModifiedAt: DateTime - дата та час останньої модифікації
- IsSaved: bool - статус збереження документа
- TextEncoding: TextEncoding - об'єкт кодування тексту

Операції:

• SetContent(content: string): void - встановлення нового контенту документа

Клас TextEncoding моделює кодування тексту для правильного відображення та збереження документів.

Атрибути:

- Id: int - унікальний ідентифікатор кодування
- Name: string - назва кодування (наприклад, "UTF-8", "UTF-16")
- CodePage: string - кодова сторінка кодування
- IsDefault: bool - чи є це кодування за замовчуванням

Операції:

• DetectFromBytes(data: byte[]): TextEncoding - автоматичне визначення кодування з байтів файлу

Клас RecentFile зберігає інформацію про нещодавно відкриті файли для швидкого доступу.

Атрибути:

- Id: int - унікальний ідентифікатор запису
- FilePath: string - повний шлях до файлу
- FileName: string - назва файлу
- LastOpenedAt: DateTime - дата та час останнього відкриття
- OpenCount: int - кількість відкриттів файлу

Операції:

• UpdateLastOpened(): void - оновлення часу останнього відкриття та збільшення лічильника

Клас EditorSettings моделює налаштування текстового редактора для персоналізації робочого середовища.

Атрибути:

- Id: int - унікальний ідентифікатор налаштувань
- FontFamily: string - сімейство шрифту (за замовчуванням "Consolas")
- FontSize: int - розмір шрифту (за замовчуванням 12)
- Theme: string - тема оформлення ("Light" або "Dark")
- WordWrap: bool - перенос рядків
- ShowLineNumbers: bool - відображення номерів рядків

Операції:

• Reset(): void - скидання налаштувань до значень за замовчуванням

Клас DocumentRepository реалізує патерн Repository для управління документами в пам'яті.

Операції:

- Add(document: Document): void - додавання нового документа

- Update(document: Document): void - оновлення існуючого документа
- Delete(id: int): void - видалення документа за ідентифікатором
- GetById(id: int): Document - отримання документа за ідентифікатором
- GetAll(): List<Document> - отримання всіх документів
- GetByPath(filePath: string): Document - пошук документа за шляхом до файлу

Клас RecentFileRepository управляє списком нещодавно відкритих файлів.

Операції:

- GetRecent(count: int): List<RecentFile> - отримання останніх файлів
- AddOrUpdate(file: RecentFile): void - додавання або оновлення запису
- Delete(id: int): void - видалення файлу зі списку

Клас EncodingRepository управляє доступними кодуваннями тексту.

Операції:

- GetDefault(): TextEncoding - отримання кодування за замовчуванням
- GetByCodePage(codePage: string): TextEncoding - пошук за кодовою сторінкою
- GetAll(): List<TextEncoding> - отримання всіх доступних кодувань

Клас EditorSettingsRepository управляє налаштуваннями редактора.

Операції:

- GetCurrent(): EditorSettings - отримання поточних налаштувань
- Update(settings: EditorSettings): void - оновлення налаштувань

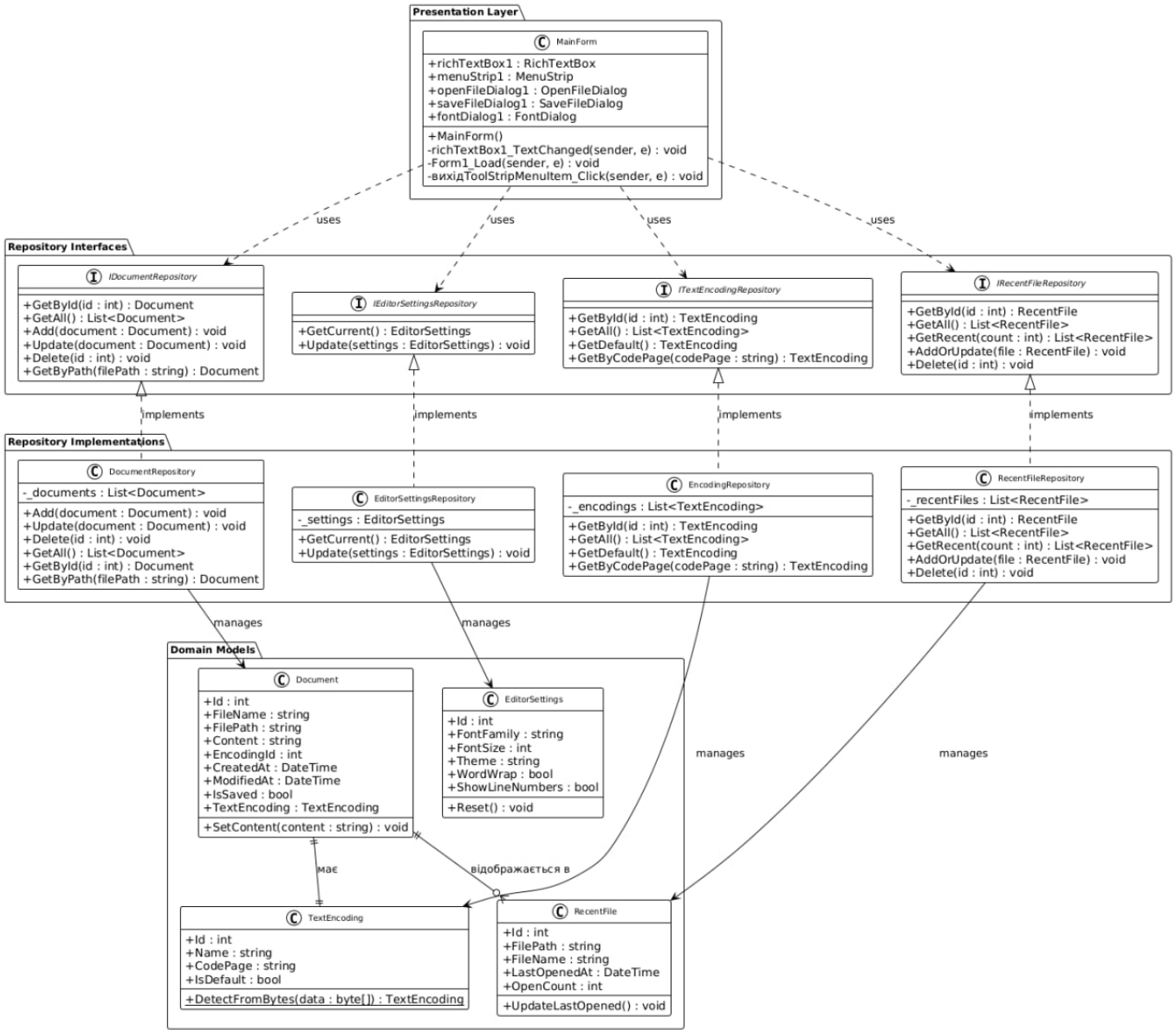
Взаємозв'язки:

• Document пов'язаний з TextEncoding (один-до-одного)

- Document може мати багато записів в RecentFile (один-до-багатьох)
- DocumentRepository управляє колекцією Document
- RecentFileRepository управляє колекцією RecentFile
- EncodingRepository забезпечує Document об'єктами TextEncoding
- EditorSettingsRepository зберігає єдиний екземпляр EditorSettings для системи

Система побудована за принципами Repository Pattern та розділення відповідальностей, що забезпечує гнучкість та можливість легкого тестування компонентів.

Діаграма класів предметної області:



Сценарій 1 - Створення нового документа

Передумови: Програма текстового редактора запущена.

Постумови: Створено новий порожній документ готовий для введення тексту.

Взаємодіючі сторони: Користувач, Система.

Короткий опис: Користувач створює новий текстовий документ для написання тексту.

Основний потік подій:

1. Користувач натискає кнопку "Новий файл" в меню.
2. Система створює порожній документ з назвою "Без назви".
3. Система очищує область для введення тексту.
4. Система встановлює кодування UTF-8 за замовчуванням.
5. Система показує готовність до введення тексту.
6. Користувач може починати друкувати текст.

Винятки: Виняток №1: Якщо попередній документ не збережений, система запитує чи зберегти його перед створенням нового.

Примітки: Новий документ отримує поточну дату створення та автоматично відслідковує зміни.

Сценарій 2 - Відкриття існуючого файлу

Передумови: На комп'ютері існує текстовий файл який потрібно відкрити.

Постумови: Файл завантажений в редактор та показаний користувачу. Файл додається до списку недавно відкритих.

Взаємодіючі сторони: Користувач, Система, Файлова система.

Короткий опис: Користувач відкриває збережений раніше текстовий файл для перегляду або редагування.

Основний потік подій:

1. Користувач натискає кнопку "Відкрити файл" в меню.
2. Система показує вікно для вибору файлу на комп'ютері.
3. Користувач обирає потрібний файл і натискає "Відкрити".
4. Система читає файл з диска та визначає його кодування.
5. Система показує зміст файлу в області редагування.
6. Система встановлює назву файлу в заголовку вікна.
7. Система додає файл до списку недавно відкритих файлів.
8. Система оновлює дату останнього відкриття файлу.

Винятки: Виняток №1: Файл не існує або пошкоджений - система повідомляє про помилку. Виняток №2: Файл занадто великий - система попереджає про можливу повільну роботу.

Примітки: Система автоматично розпізнає кодування тексту для правильного відображення символів.

Сценарій 3 - Збереження документа

Передумови: Користувач ввів або змінив текст в документі.

Постумови: Документ збережений на диск з усіма змінами. Статус документа помічений як збережений.

Взаємодіючі сторони: Користувач, Система, Файлова система.

Короткий опис: Користувач зберігає введений або змінений текст у файл на комп'ютері.

Основний потік подій:

1. Користувач натискає кнопку "Зберегти" в меню або натискає Ctrl+S.

2. Система перевіряє чи документ має назву файлу.

3. Якщо назва є - система зберігає файл за існуючим шляхом.

4. Якщо назви немає - система показує вікно "Зберегти як" для вибору місця та назви.

5. Користувач вказує де зберегти файл та як його назвати.

6. Система записує весь текст у файл з правильним кодуванням.

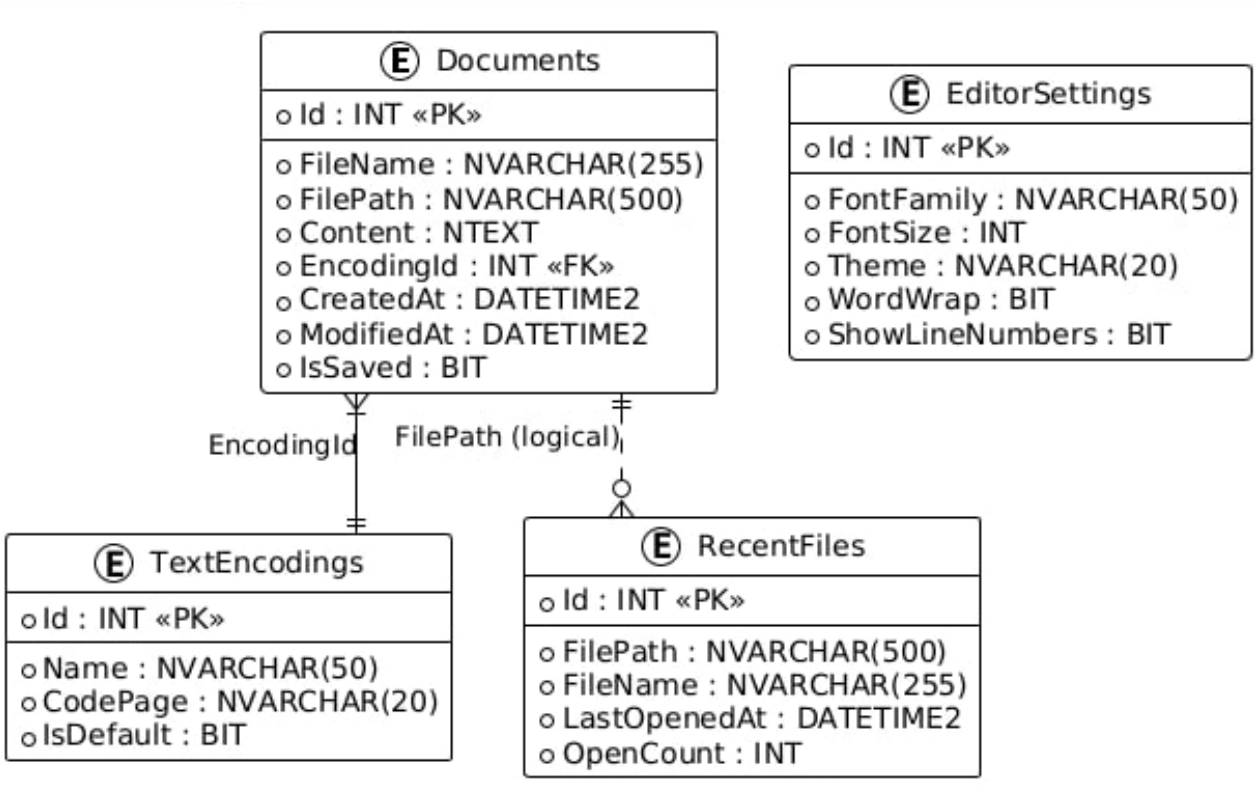
7. Система позначає документ як збережений.

8. Система оновлює час останньої модифікації.

9. Система показує повідомлення про успішне збереження.

Винятки: Виняток №1: Недостатньо місця на диску - система повідомляє про помилку. Виняток №2: Немає прав для запису в обрану папку - система пропонує інше місце. Виняток №3: Файл використовується іншою програмою - система пропонує зберегти під іншою назвою.

Структура бази даних:



Код реалізації:

```
using System;
using System.Text;
using TextEditorMK.Models;

namespace TextEditorMK.Models
{
    13 references
    public class Document
    {
        3 references
        public int Id { get; set; }
        6 references
        public string FileName { get; set; } = string.Empty;
        8 references
        public string FilePath { get; set; } = string.Empty;
        5 references
        public string Content { get; set; } = string.Empty;
        3 references
        public int EncodingId { get; set; }
        0 references
        public DateTime CreatedAt { get; set; } = DateTime.Now;
        3 references
        public DateTime ModifiedAt { get; set; } = DateTime.Now;
        3 references
        public bool IsSaved { get; set; } = false;

        1 reference
        public TextEncoding TextEncoding { get; set; }

        1 reference
        public void SetContent(string content)
        {
            Content = content;
            ModifiedAt = DateTime.Now;
            IsSaved = false;
        }
    }
}
```

Document.cs

```
namespace TextEditorMK.Models
{
    6 references
    public class EditorSettings
    {
        0 references
        public int Id { get; set; }
        1 reference
        public string FontFamily { get; set; } = "Consolas";
        1 reference
        public int FontSize { get; set; } = 12;
        3 references
        public string Theme { get; set; } = "Light";
        1 reference
        public bool WordWrap { get; set; } = false;
        1 reference
        public bool ShowLineNumbers { get; set; } = true;

        0 references
        public void Reset()
        {
            FontFamily = "Consolas";
            FontSize = 12;
            Theme = "Light";
            WordWrap = false;
            ShowLineNumbers = true;
        }
    }
}
```

EditorSettings.cs

```
using System;

namespace TextEditorMK.Models
{
    11 references
    public class RecentFile
    {
        2 references
        public int Id { get; set; }
        3 references
        public string FilePath { get; set; } = string.Empty;
        2 references
        public string FileName { get; set; } = string.Empty;
        3 references
        public DateTime LastOpenedAt { get; set; } = DateTime.Now;
        1 reference
        public int OpenCount { get; set; } = 0;

        1 reference
        public void UpdateLastOpened()
        {
            LastOpenedAt = DateTime.Now;
            OpenCount++;
        }
    }
}
```

RecentFile.cs

```
namespace TextEditorMK.Models
{
    17 references
    public class TextEncoding
    {
        3 references
        public int Id { get; set; }
        5 references
        public string Name { get; set; } = "UTF-8";
        6 references
        public string CodePage { get; set; } = "utf-8";
        4 references
        public bool IsDefault { get; set; } = false;

        0 references
        public static TextEncoding DetectFromBytes(byte[] data)
        {
            if (data.Length >= 3 && data[0] == 0xEF && data[1] == 0xBB && data[2] == 0xBF)
                return new TextEncoding { Name = "UTF-8", CodePage = "utf-8" };

            if (data.Length >= 2 && data[0] == 0xFF && data[1] == 0xFE)
                return new TextEncoding { Name = "UTF-16 LE", CodePage = "utf-16" };

            return new TextEncoding { Name = "UTF-8", CodePage = "utf-8", IsDefault = true };
        }
    }
}
```

TextEncoding.cs

```cs
using System.Collections.Generic;
using System.Linq;

using TextEditorMK.Models;
using TextEditorMK.Repositories.Interfaces;

namespace TextEditorMK.Repositories.Implementations
{
    1 reference
    public class DocumentRepository : IDocumentRepository
    {
        private readonly List<Document> _documents = new List<Document>();

        2 references
        public void Add(Document document) => _documents.Add(document);

        2 references
        public void Update(Document document)
        {
            var existing = GetById(document.Id);
            if (existing != null)
            {
                existing.FileName = document.FileName;
                existing.FilePath = document.FilePath;
                existing.Content = document.Content;
                existing.EncodingId = document.EncodingId;
                existing.ModifiedAt = document.ModifiedAt;
                existing.IsSaved = document.IsSaved;
            }
        }

        1 reference
        public void Delete(int id)
        {
            var doc = GetById(id);
            if (doc != null) _documents.Remove(doc);
        }

        1 reference
        public List<Document> GetAll() => _documents;

        3 references
        public Document GetById(int id) => _documents.FirstOrDefault(d => d.Id == id);

        1 reference
        public Document GetByPath(string filePath) =>
            _documents.FirstOrDefault(d => d.FilePath == filePath);
    }
}
```

DocumentRepository.cs

```cs
using TextEditorMK.Models;
using TextEditorMK.Repositories.Interfaces;

namespace TextEditorMK.Repositories.Implementations
{
    1 reference
    public class EditorSettingsRepository : IEditorSettingsRepository
    {
        private EditorSettings _settings = new EditorSettings();

        2 references
        public EditorSettings GetCurrent() => _settings;

        2 references
        public void Update(EditorSettings settings)
        {
            _settings = settings;
        }
    }
}
```

EditorSettingsRepository.cs

```cs
using System.Collections.Generic;
using System.Linq;

using TextEditorMK.Repositories.Interfaces;
using TextEditorMK.Models;

namespace TextEditorMK.Repositories.Implementations
{
    1 reference
    public class EncodingRepository : IEncodingRepository
    {
        private readonly List<TextEncoding> _encodings = new List<TextEncoding>()
        {
            new TextEncoding { Id = 1, Name = "UTF-8", CodePage = "utf-8", IsDefault = true },
            new TextEncoding { Id = 2, Name = "UTF-16 LE", CodePage = "utf-16", IsDefault = false }
        };

        1 reference
        public TextEncoding GetById(int id) => _encodings.FirstOrDefault(e => e.Id == id);

        1 reference
        public List<TextEncoding> GetAll() => _encodings;

        2 references
        public TextEncoding GetDefault() => _encodings.FirstOrDefault(e => e.IsDefault);

        1 reference
        public TextEncoding GetByCodePage(string codePage) =>
            _encodings.FirstOrDefault(e => e.CodePage == codePage);
    }
}
```

EncodingRepository.cs

```cs
using System.Collections.Generic;
using System.Linq;

using TextEditorMK.Models;
using TextEditorMK.Repositories.Interfaces;

namespace TextEditorMK.Repositories.Implementations
{

    public class RecentFileRepository : IRecentFileRepository
    {
        private readonly List<RecentFile> _recentFiles = new List<RecentFile>();

        2 references
        public RecentFile GetById(int id) => _recentFiles.FirstOrDefault(r => r.Id == id);

        1 reference
        public List<RecentFile> GetAll() => _recentFiles;

        2 references
        public List<RecentFile> GetRecent(int count) =>
            _recentFiles.OrderByDescending(r => r.LastOpenedAt).Take(count).ToList();

        2 references
        public void AddOrUpdate(RecentFile file)
        {
            var existing = _recentFiles.FirstOrDefault(r => r.FilePath == file.FilePath);
            if (existing == null)
            {
                _recentFiles.Add(file);
            }
            else
            {
                existing.UpdateLastOpened();
            }
        }

        1 reference
        public void Delete(int id)
        {
            var file = GetById(id);
            if (file != null) _recentFiles.Remove(file);
        }
    }
}
```

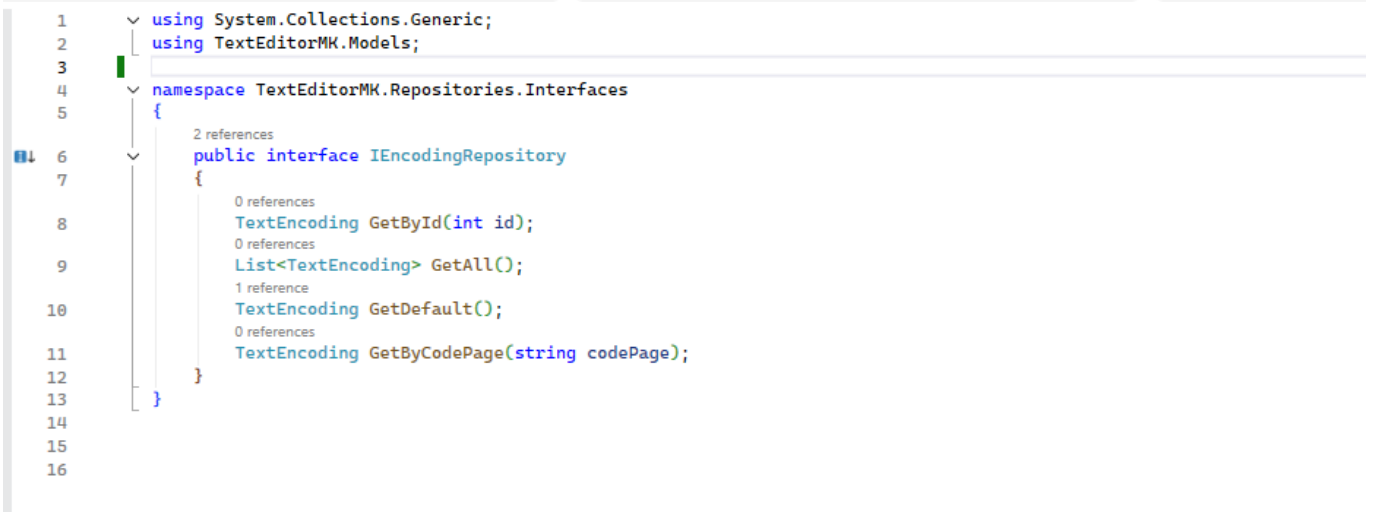
RecentFileRepository.cs

```
1   using System.Collections.Generic;
2   using TextEditorMK.Models;
3
4   namespace TextEditorMK.Repositories.Interfaces
5   {
        2 references
6       public interface IDocumentRepository
7       {
            3 references
8           Document GetById(int id);
            1 reference
9           List<Document> GetAll();
            2 references
10          void Add(Document document);
            2 references
11          void Update(Document document);
            1 reference
12          void Delete(int id);
            1 reference
13          Document GetByPath(string filePath);
14      }
15  }
16
17
18
```

IDocumentRepository.cs

```
1   using TextEditorMK.Models;
2
3   namespace TextEditorMK.Repositories.Interfaces
4   {
5       public interface IEditorSettingsRepository
6       {
7           EditorSettings GetCurrent();
8           void Update(EditorSettings settings);
9       }
10  }
11
```

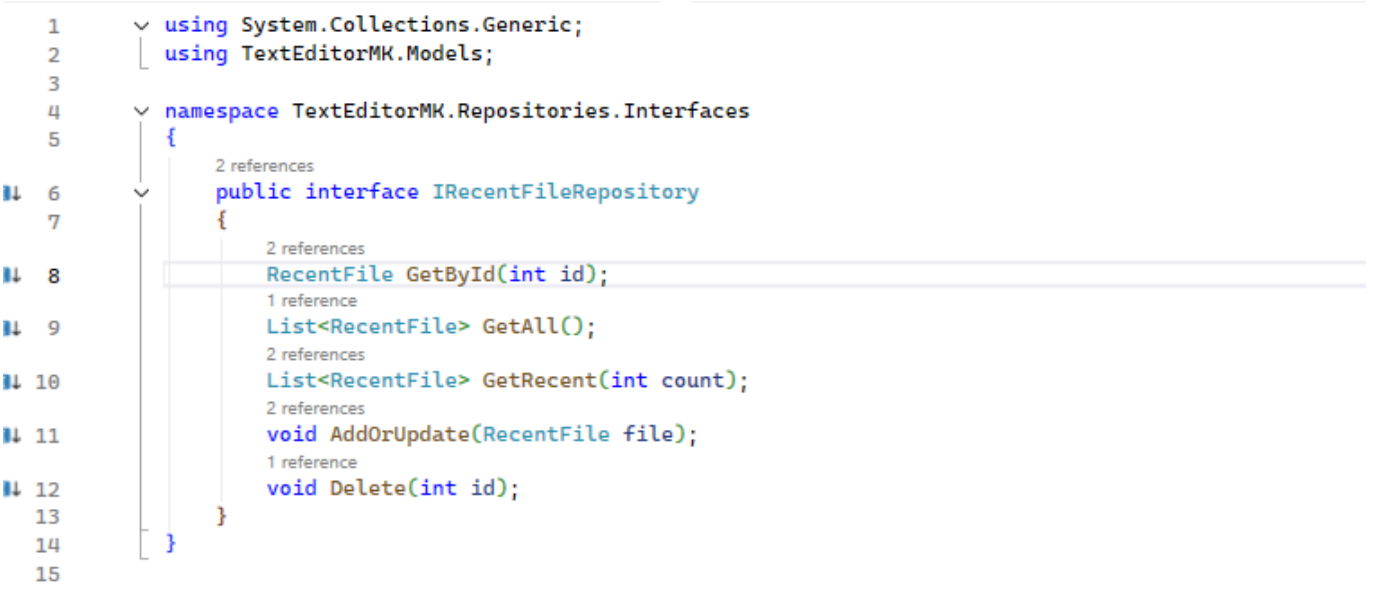
IEditorSettingsRepository.cs



```
using System.Collections.Generic;
using TextEditorMK.Models;

namespace TextEditorMK.Repositories.Interfaces
{
    2 references
    public interface IEncodingRepository
    {
        0 references
        TextEncoding GetById(int id);
        0 references
        List<TextEncoding> GetAll();
        1 reference
        TextEncoding GetDefault();
        0 references
        TextEncoding GetByCodePage(string codePage);
    }
}
```

IEncodingRepository.cs

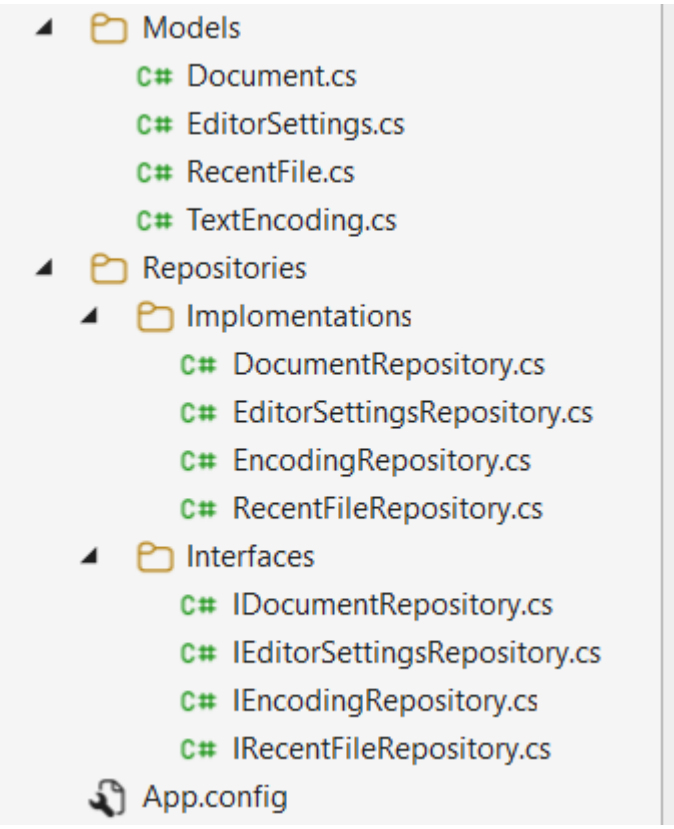


```
using System.Collections.Generic;
using TextEditorMK.Models;

namespace TextEditorMK.Repositories.Interfaces
{
    2 references
    public interface IRecentFileRepository
    {
        2 references
        RecentFile GetById(int id);
        1 reference
        List<RecentFile> GetAll();
        2 references
        List<RecentFile> GetRecent(int count);
        2 references
        void AddOrUpdate(RecentFile file);
        1 reference
        void Delete(int id);
    }
}
```

IRecentFileRepository.cs

Реалізовані класи:



Висновок: У межах лабораторної роботи я спроєктував систему Text Editor MK, яка дозволяє користувачам працювати з текстовими документами. Були створені діаграми варіантів використання та класів, розроблені сценарії використання для трьох основних функцій системи: створення документа, відкриття файлу та збереження документа. Також була спроєктована структура бази даних з чотирма таблицями (Documents, TextEncodings, RecentFiles, EditorSettings) та реалізовані основні класи з використанням шаблону Repository для ефективної взаємодії з даними. Система передбачає базовий функціонал для роботи з текстом: автоматичне визначення кодування, ведення списку нещодавніх файлів та налаштування редактора. Використано архітектурні патерни Repository, Strategy та Observer для забезпечення гнучкості та розширюваності системи.

Відповіді на контрольні питання:

1. Що таке UML? UML (Unified Modeling Language) — це уніфікована мова моделювання, яка використовується для графічного представлення, документування та проектування програмних систем. Вона допомагає описати структуру, поведінку та взаємодію компонентів системи.

2. Що таке діаграма класів UML?

Діаграма класів UML — це статична діаграма, яка показує класи системи, їх атрибути, методи та зв'язки між класами. Вона використовується для моделювання структури об'єктно-орієнтованих програм.

3. Які діаграми UML називають канонічними?

Канонічні (основні) діаграми UML — це ті, що найчастіше використовуються для опису системи:

• Діаграма класів

• Діаграма варіантів використання (Use Case)

• Діаграма послідовності (Sequence)

• Діаграма станів (State)

• Діаграма активностей (Activity)

4. Що таке діаграма варіантів використання?

Діаграма варіантів використання (Use Case Diagram) показує взаємодію користувачів (акторів) з системою через варіанти використання. Вона відображає, хто і як використовує систему, без деталізації внутрішньої реалізації.

5. Що таке варіант використання?

Варіант використання (Use Case) — це послідовність дій, які система виконує для досягнення певної мети користувача. Наприклад, “Реєстрація користувача” або “Надіслати повідомлення”.

6. Які відношення можуть бути відображені на діаграмі використання?

На діаграмі варіантів використання можуть бути:

• Association (асоціація) — зв’язок між актором і варіантом використання.

• Include (включення) — один варіант використання завжди виконує інший.

• Extend (розширення) — додатковий варіант, який може виконуватися у певних умовах.

• Generalization (успадкування) — актор або варіант використання наслідує інший.

7. Що таке сценарій?

Сценарій — це конкретна реалізація варіанту використання, тобто послідовність кроків, які відбуваються під час виконання дії користувачем та системою.

8. Що таке діаграма класів?

Діаграма класів — це графічне представлення класів, їх атрибутів і методів, а також зв’язків між класами. Вона описує структуру системи.

9. Які зв’язки між класами ви знаєте?

Основні типи зв’язків:

• Асоціація (Association) — загальний зв’язок між класами.

• Агрегація (Aggregation) — «має»; клас складається з інших, але частини можуть існувати окремо.

• Композиція (Composition) — сильніша агрегація; частини не можуть існувати

без цілого.

• Успадкування (Generalization) — один клас наслідує властивості іншого.

• Залежність (Dependency) — один клас використовує інший тимчасово.

10. Чим відрізняється композиція від агрегації?

• Агрегація: частини можуть існувати без цілого.

• Композиція: частини не можуть існувати без цілого; знищення цілого знищує частини.

11. Чим відрізняється агрегація від композиції на діаграмах класів?

• Агрегація позначається порожнім ромбом на стороні цілого.

• Композиція позначається заповненим ромбом на стороні цілого.

12. Що являють собою нормальні форми баз даних?

Нормальні форми — це правила організації таблиць БД для уникнення надлишковості та аномалій при оновленні даних. Основні:

• 1НФ — всі атрибути атомарні.

• 2НФ — всі неключові атрибути залежать від усього первинного ключа.

• 3НФ — немає транзитивних залежностей між неключовими атрибутами.

13. Що таке фізична і логічна модель БД?

• Логічна модель — структура даних з таблицями, полями та зв’язками, незалежно від СУБД.

• Фізична модель — конкретна реалізація БД у СУБД, включає типи даних, індекси, обмеження, фізичне зберігання.

14. Який взаємозв'язок між таблицями БД та програмними класами?

Кожна таблиця БД часто відповідає класу у програмі, а рядки таблиці — об’єктам класу. Поля таблиці відображаються як атрибути класу, а зв’язки між

таблицями як зв’язки між класами.