Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №2

Технологія розробки програмного забезпечення

«ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ. СЦЕНАРІЇ ВАРІАНТІВ ВИКОРИСТАННЯ. ДІАГРАМИ UML. ДІАГРАМИ КЛАСІВ. КОНЦЕПТУАЛЬНА МОДЕЛЬ СИСТЕМИ»

Виконала студентка
групи ІА-23:
Кашуб’як С. М.

Перевірив:
Мягкий М. Ю.

Київ 202

Зміст

Тема: Діаграма варіантів використання. Сценарії варіантів використання. Діаграми UML. Діаграми класів. Концептуальна модель системи

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.
2. Проаналізуйте тему та намалюйте схему прецеденту, що відповідає обраній темі лабораторії.
3. Намалюйте діаграму класів для реалізованої частини системи.
4. Виберіть 3 прецеденти і напишіть на їх основі прецеденти.
5. Розробити основні класи і структуру системи баз даних.
6. Класи даних повинні реалізувати шаблон Репозиторію для взаємодії з базою даних.
7. Підготувати звіт про хід виконання лабораторних робіт. Звіт, що подається повинен містити: діаграму прецедентів, діаграму класів системи, вихідні коди класів системи, а також зображення структури бази даних.

Варіант:

..14 Архіватор (strategy, adapter, factory method, facade, visitor, p2p)

Архіватор повинен являти собою візуальний додаток з можливістю створення і редагування архівів різного типу (.tar.gz, .zip, .rar, .ace) додавання/ видалення файлів / папок, редагування метаданих (по можливості), перевірка checksum архівів, тестування архівів на наявність пошкоджень, розбиття архівів на частини.

Теоретичні відомості

• Діаграма варіантів використання (Use Case Diagram) – це тип діаграми UML, що описує функціональність системи з точки зору її користувачів (акторів) і взаємодії між ними та системою. Вона показує, які дії (варіанти використання) можуть виконуватися користувачами, але не вдається у внутрішні механізми їх реалізації.

• Сценарії варіантів використання (Use Case Scenarios) – це текстовий опис варіантів використання, де детально викладається, як система повинна реагувати на дії користувачів у кожній конкретній ситуації. Включає в себе основний потік подій та альтернативні шляхи розвитку сценарію.

• Діаграма класів (Class Diagram) – це структура, яка моделює класи системи, їх властивості, методи, а також зв'язки між ними. Класи представляють основні об'єкти системи, які мають атрибути та операції, а також відображають взаємодію між різними компонентами.

• Концептуальна модель системи – це абстрактне представлення об'єктів та зв'язків між ними, що відображає ключові аспекти системи з точки зору бізнесу або

предметної області. Вона описує основні компоненти, їх взаємодію та структуру, але не деталізує технічну реалізацію. Ці діаграми дозволяють аналізувати вимоги до системи та планувати її розробку.

Хід Роботи

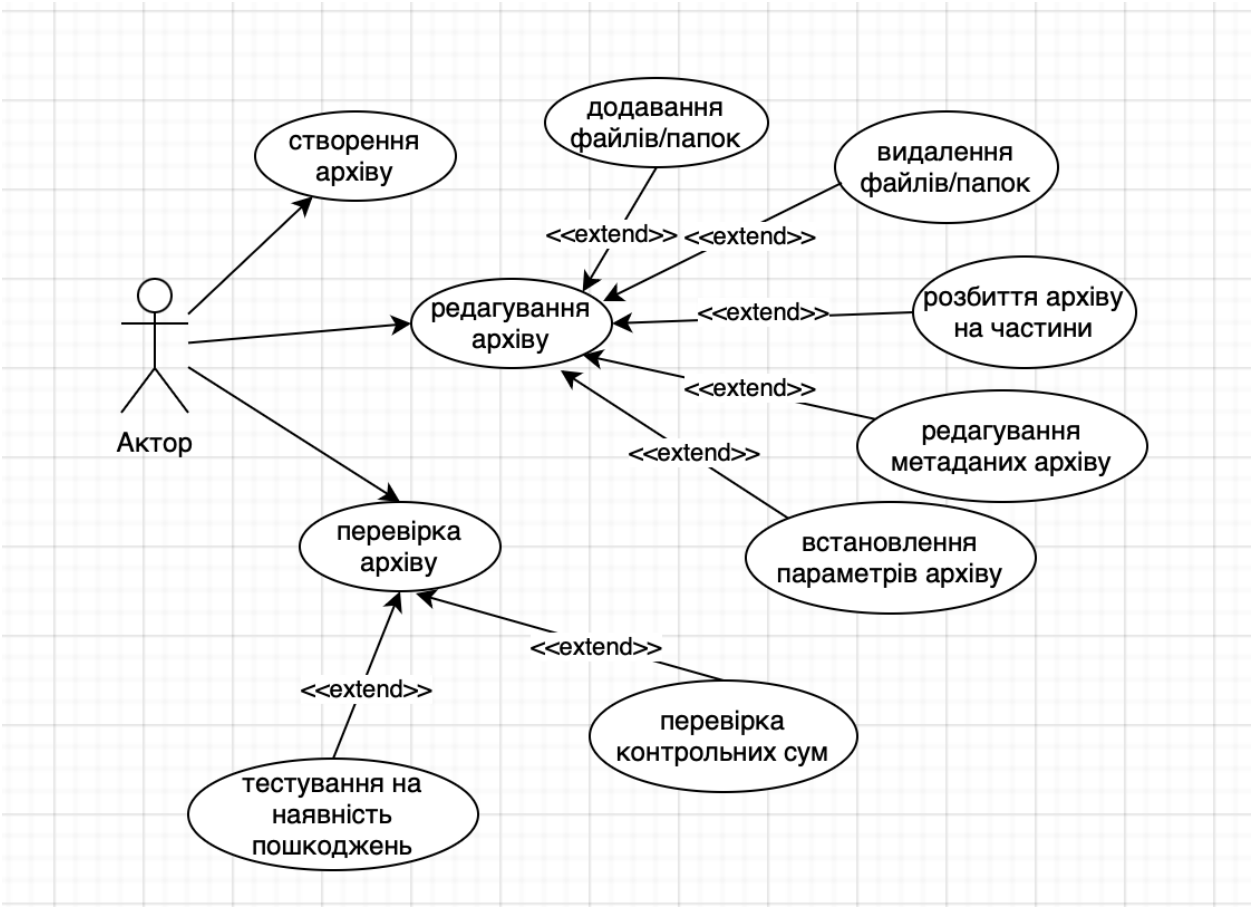

Рис 1. Діаграма прецендентів

Діаграма прецендентів

Користувач запускає архіватор та може:

- Створити архів
- Редагувати архів (додавати та видаляти файли/папки, розбивати архів на частини, редагувати метадані архіву, встановлювати параметри архіву)
- Перевіряти та тестувати архів (перевіряти контрольні суми, тестувати на наявність пошкоджень)

Предеценти на основі трьох прецедентів:

Прецедент 1: Створення архіву

- Передумови: користувач має доступ до необхідних файлів і програми архіватора для створення архіву.

- Постумови: система створює архів на основі вибраних файлів і параметрів. Якщо виникають помилки, система повідомляє користувача про проблему, і архів не створюється.
- Сторони взаємодії: користувач, система архіватора.
- Короткий опис: цей варіант використання описує процес створення архіву з вибраними файлами та зазначеним типом архіву.

Основний потік подій:

1. Користувач відкриває архіватор.
2. Користувач вибирає опцію створення нового архіву.
3. Система запитує вибір типу архіву (наприклад, .zip, .tar.gz, .rar).
4. Користувач вибирає тип архіву та додає файли/папки для архівації.
5. Система підтверджує вибір файлів і параметрів архіву.
6. Користувач вибирає місце для збереження архіву і натискає кнопку "Створити".
7. Система створює архів на основі вибраних файлів та параметрів.
8. Після завершення створення архіву система повідомляє користувача про успішне завершення операції або виникнення помилок.

Винятки:

- Виняток №1: невірно вибрані файли або некоректні параметри архіву. Якщо система не може створити архів через помилки в даних, вона повідомляє про це користувача та пропонує виправити введені дані або скасувати процес.
- Виняток №2: немає вільного місця на диску для збереження архіву. Система повідомляє про це користувача і пропонує вибрати інше місце для збереження архіву.

Прецедент 2: Редагування архіву

- Передумови: користувач має доступ до існуючого архіву, який потрібно відредагувати (додати або видалити файли, змінити метадані, налаштувати параметри архіву).
- Постумови: система успішно редагує архів на основі введених користувачем змін. Якщо виникають помилки, система повідомляє користувача про проблему, і архів не редагується.
- Сторони взаємодії: користувач, система архіватора.

- Короткий опис: цей варіант використання описує процес редагування існуючого архіву, зокрема додавання або видалення файлів, редагування метаданих та налаштування параметрів архіву.

Основний потік подій:

1. Користувач відкриває архіватор.
2. Користувач вибирає архів, який потрібно відредагувати.
3. Система запитує, які зміни необхідно внести (додавання файлів/папок, видалення файлів, редагування метаданих тощо).
4. Користувач вибирає відповідні операції (додавання файлів/папок, видалення, зміна метаданих).
5. Система підтверджує вибір змін і запитує місце для збереження зміненого архіву.
6. Користувач вказує місце для збереження редагованого архіву або вибирає заміну існуючого архіву.
7. Система вносить зміни до архіву.
8. Після завершення редагування архіву система повідомляє користувача про успішне завершення операції або виникнення помилок.

Винятки:

- Виняток №1: невірний формат файлів або помилки при додаванні файлів. Якщо система не може додати файли через некоректний формат, вона повідомляє користувача про помилку і пропонує виправити дані.
- Виняток №2: пошкодження архіву при редагуванні. Якщо архів пошкоджений під час редагування, система повідомляє користувача про помилку і пропонує спробувати відновити архів.
- Виняток №3: недостатньо вільного місця на диску для збереження змін. Якщо немає достатньо місця для збереження редагованого архіву, система повідомляє про це користувача і пропонує вибрати інший диск або місце для збереження.

Прецедент 3: Перевірка та тестування архіву
- Передумови: користувач має доступ до архіву, який необхідно перевірити на наявність помилок або пошкоджень. Архів має бути доступний для перевірки контрольних сум або тестування на пошкодження.
- Постумови: система успішно перевіряє архів, надаючи результат тестування (контрольні суми, наявність пошкоджень). Якщо виникають помилки, система повідомляє користувача про проблему.
- Сторони взаємодії: користувач, система архіватора.

- Короткий опис: цей варіант використання описує процес перевірки архіву на наявність помилок, таких як пошкодження або невідповідність контрольних сум.

Основний потік подій:

1. Користувач відкриває архіватор.
2. Користувач вибирає архів, який потрібно перевірити.
3. Система пропонує кілька варіантів перевірки: контрольні суми, тестування на наявність пошкоджень.
4. Користувач вибирає опцію перевірки архіву (перевірка контрольних сум або тестування на пошкодження).
5. Система виконує вибрану перевірку (перевіряє контрольні суми або тестує архів на пошкодження).
6. Після завершення перевірки система надає результат користувачу (успішно пройшло тестування або виявлені помилки).
7. Якщо є помилки, система повідомляє користувача і пропонує можливі дії (відновлення архіву, спроба перевірки знову, або інше).

Винятки:

- Виняток №1: невідповідність контрольних сум. Якщо система виявляє невідповідність контрольної суми архіву, вона повідомляє про це користувача і пропонує перевірити файл чи повторно створити архів.
- Виняток №2: пошкодження архіву. Якщо архів пошкоджений і не може бути перевірений, система надає повідомлення про пошкодження архіву та пропонує відновлення або інші варіанти.
- Виняток №3: відсутність файлів або доступу до архіву. Якщо архів неможливо відкрити або немає доступу до необхідних файлів для тестування, система повідомляє про проблему з доступом або наявністю файлів і пропонує інші варіанти дій.

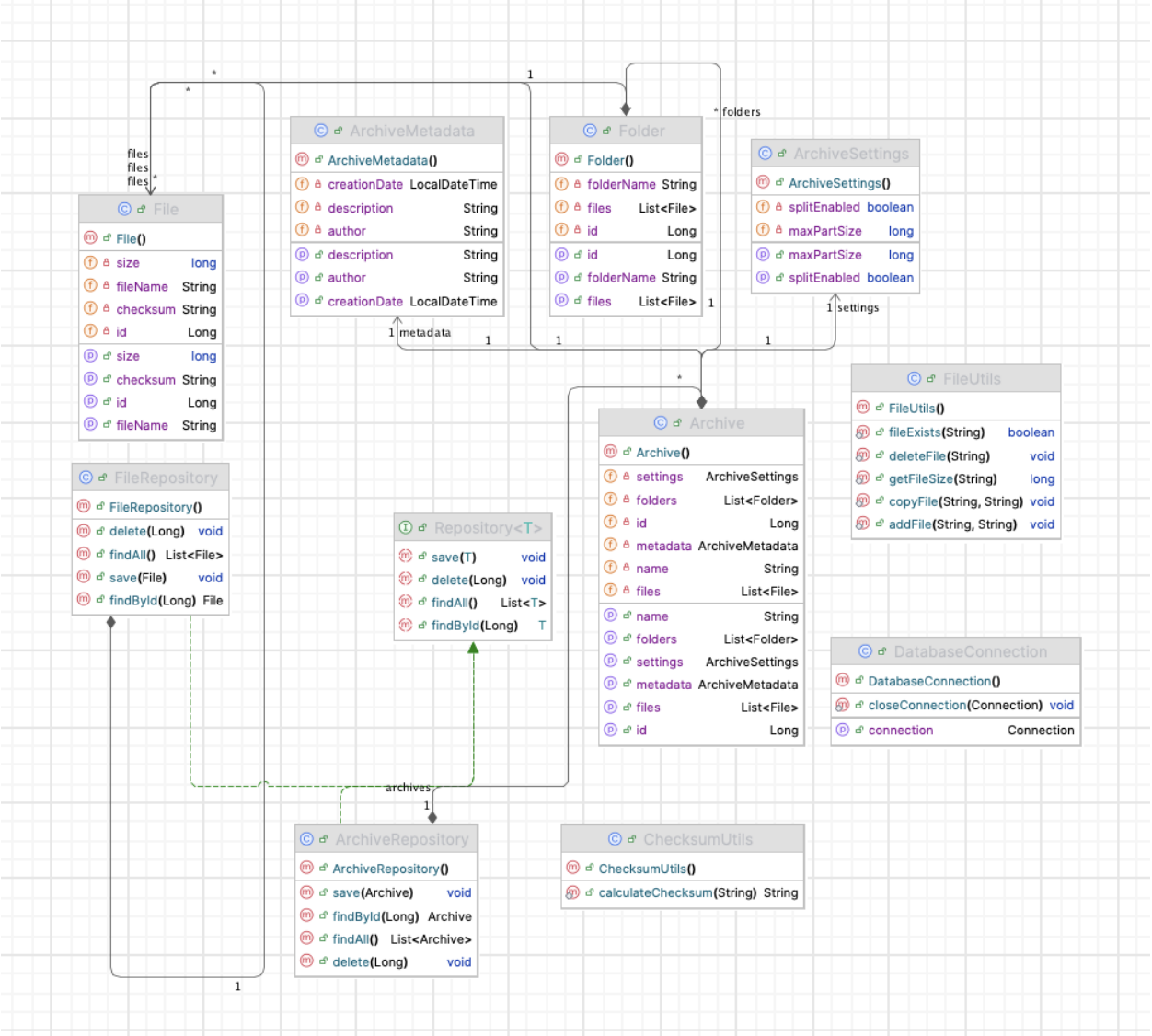

Рис 2. Діаграма класів

Діаграма класів

Основні компоненти діаграми:

1. Repository Pattern:

- **Repository<T>** – це базовий інтерфейс, який визначає загальні CRUD-операції для всіх моделей: ‘save(T t)’ - для збереження об'єкта, ‘findById(Long id)’ - для пошуку об'єкта за ідентифікатором, ‘delete(Long id)’ - для видалення об'єкта, ‘findAll()’ - для отримання списку всіх об'єктів. Цей інтерфейс забезпечує розділення логіки роботи з базою даних та логіки додатка.

2. **Моделі:**

- **File** – модель, яка описує файл, з такими полями: ‘id’ (Long) -ідентифікатор файл, ‘fileName’ (String) - назва файлу, ‘size’ (long) - розмір файлу, ‘checksum’ (String) - контрольна сума файлу. Цей клас пов'язаний з іншими моделями, наприклад, архівами.
- **Folder** – модель для опису папки. Поля: ‘id’ (Long) - ідентифікатор папки, ‘folderName’ (String) - назва папки, ‘files’ (List<File>) - список файлів у папці.
- **Archive** – основна модель для опису архіву. Поля: ‘id’ (Long) - ідентифікатор архіву, ‘name’ (String) - назва архіву, ‘metadata’ (ArchiveMetadata) - метадані архіву. ‘settings’ (ArchiveSettings) - налаштування архіву, ‘folders’ (List<Folder>) - список папок в архіві, ‘files’ (List<File>) -список файлів в архіві. Архів пов'язаний із кількома іншими моделями, такими як ‘Folder’, ‘File’, ‘ArchiveSettings’.
- **ArchiveMetadata** – модель для збереження метаданих архіву. Поля: ‘creationDate’ (LocalDateTime) - дата створення архіву, ‘description’ (String) - опис архіву, ‘author’ (String) - автор архіву.
- **ArchiveSettings** – модель, що описує налаштування архіву. Поля: ‘maxPartSize’ (long) - максимальний розмір частини архіву, ‘splitEnabled’ (boolean) - чи увімкнено поділ архіву на частини.

3. **Зв'язки між класами:**

- **‘File’** та **‘Folder’** – кожен об'єкт ‘Folder’ містить список файлів (‘files’), пов'язаних з ним.
- **‘Archive’**, **‘Folder’**, та **‘File’** – об'єкт ‘Archive’ має список папок (‘folders’) і список файлів (‘files’).
- **‘Archive’** та ‘**ArchiveMetadata’** – кожен архів має свої метадані, які описують основні властивості архіву.
- **‘Archive’** та **‘ArchiveSettings’** – налаштування архіву визначають параметри його створення, такі як розмір частин і поділ.

4. **Репозиторії:**

- **FileRepository** – забезпечує CRUD-операції для моделі ‘File’, включає методи ‘save’, ‘findById’, ‘delete’, та ‘findAll’.
- **ArchiveRepository** – забезпечує CRUD-операції для моделі ‘Archive’, додатково може містити специфічну логіку для роботи з архівами (наприклад, пошук за назвою).

- **Базовий Repository<T>** – використовується як шаблон для створення спеціалізованих репозиторіїв.

5. **Utility-класи:**

- **FileUtils** – допоміжний клас для роботи з файлами. Методи: ‘fileExists(String path)’ - перевіряє, чи існує файл за вказаним шляхом, ‘addFile(String source, String destination)’ - додає файл, ‘deleteFile(String path)’ - видаляє файл, ‘copyFile(String source, String destination)’ - копіює файл, ‘getFileSize(String path)’ - отримує розмір файлу.
- **ChecksumUtils** – клас для обчислення контрольної суми файлів. ‘calculateChecksum(String path)’ - обчислює контрольну суму для файлу.

6. **База даних та з'єднання:**

- **DatabaseConnection** – клас для управління з'єднанням з базою даних. Методи: ‘getConnection()’ - встановлює та повертає з'єднання з базою, ‘closeConnection(Connection connection)’ - закриває активне з'єднання.

Загальна структура:

- **Моделі** – відображають реальні об'єкти у програмі (файли, папки, архіви, метадані).
- **Репозиторії** – забезпечують CRUD-операції для кожної з моделей, зберігаючи при цьому розділення логіки доступу до даних і бізнес-логіки.
- **Utility-класи** – забезпечують виконання операцій з файлами та перевірку контрольних сум.

Структура бази даних

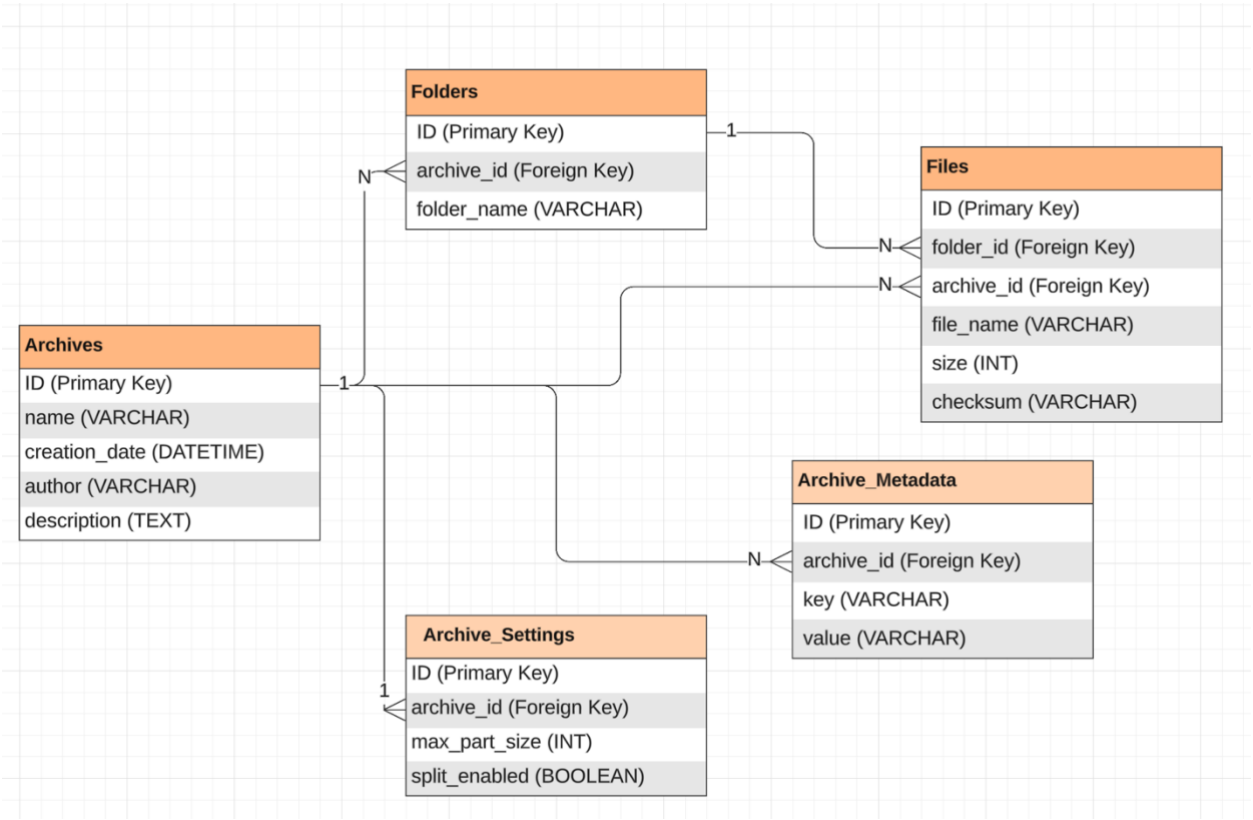

Рис 3. Структура бази даних

Висновок: отже, у ході виконання лабораторної роботи було проведено ознайомлення з теоретичними відомостями та розроблено прецеденти та діаграми класів для системи керування завданнями. Окрім того, підготовлений звіт включає всі необхідні компоненти, що відображають структуру розробленої системи.