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
«Основи проектування»

Виконала студентка
групи ІА–32:
Ткачук М. С.

Перевірив:
Мягкий М. Ю.

Київ 2025

Зміст

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 реалізувати шаблон 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

Аудіо редактор (singleton, adapter, observer, mediator, composite, client-server)

Аудіо редактор повинен володіти наступним функціоналом: представлення аудіо даних будь-якого формату в WAVE-формі, вибір і подальші операції копіювання / вставки / вирізання / деформації по сегменту аудіозапису, можливість роботи з декількома звуковими доріжками, кодування в найбільш поширених форматах (ogg, flac, mp3).

Теоретичні відомості

UML (Unified Modeling Language) — універсальна мова візуального моделювання для специфікації, візуалізації, проєктування й документування програмних систем та бізнес-процесів.

UML дозволяє створювати концептуальні, логічні та фізичні моделі складних систем на різних рівнях абстракції.

В ООАП (об’єктно-орієнтований аналіз і проєктування) модель системи складається з різних уявлень, які відображають її поведінку або структуру.

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 4 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

Прецедент 1: Додавання аудіо

Передумови:

- Користувач запустив аудіоредактор.
- Створено або відкрито проєкт.
- Користувач має локальні аудіофайли.

Постумови:

- Аудіофайл додано до проєкту як кліп/доріжку.
- Автоматично формується WAVE-подання.

Сторони взаємодії:

Користувач, система аудіоредактора.

Короткий опис:

Користувач додає аудіофайл у проєкт для подальшої обробки, після чого система генерує WAVE-подання.

Основний потік подій:

1. Користувач обирає дію «Додати».
2. Система відкриває діалог вибору файлу.
3. Користувач вибирає файл і підтверджує.
4. Система додає файл у проєкт і створює WAVE-візуалізацію.
5. Файл з’являється на таймлайні.

Винятки:

- Непідтримуваний формат: система повідомляє про помилку.
- Файл пошкоджений: імпорт скасовується.
- Відсутній доступ до файлу: система просить надати доступ.
- Нестача ресурсів: система пропонує звільнити пам’ять.

Прецедент 2: Редагування аудіо

Передумови:

- У проєкті є додані кліпи.
- Користувач бачить WAVE-подання на таймлайні.

Постумови:

- Зміни застосовано до проєкту (копіювання/вставлення/вирізання/деформація).
- WAVE-подання оновлене відповідно до змін.

Сторони:

Користувач, аудіоредактор.

Короткий опис:

Користувач виділяє ділянку аудіо та виконує операції редагування.

Основний потік подій:

1. Користувач обирає прецедент «Редагувати».
2. Виділяє сегмент на таймлайні.
3. Обирає дію:
    - Копіювати
    - Вставити
    - Вирізати
    - Стиснути/розтягнути (деформація)
4. Система застосовує вибрану операцію.
5. Оновлюється відображення таймлайна.
6. Користувач переглядає результат та при потребі повторює дії.

Винятки:

- Сегмент не виділено: система просить задати межі.
- Вставка у заблоковану зону: пропонує інше місце або заміну.
- Неприпустима деформація: система попереджає про артефакти.
- Вирізання неможливе (наприклад, в системному кліпі) — відображається помилка.

Прецедент 3: Представлення у WAVE-формі

Передумови:

- У проєкт додано хоча б один аудіофайл

Постумови:

- Хвильова форма відображається коректно на таймлайні.

Короткий опис:

Система аналізує аудіо та генерує його WAVE-подання для зручності редагування.

Основний потік:

1. Після додавання файлу система зчитує його дані.
2. Створює масив значень амплітуди.
3. Відображає WAVE-графік на таймлайні.
4. Оновлює його при кожній операції редагування.

Винятки:

- Неможливо побудувати хвильову форму (файл пошкоджений) система повідомляє користувача.
- Формат надто великий система спрощує візуалізацію.

Прецедент 4: Збереження

Передумови:

- У проєкті є хоча б один кліп.
- Користувач готовий зберегти проєкт або фінальний файл.

Постумови:

- Створено файл у форматі MP3 / OGG / FLAC (через <<extend>>).

- Система повідомляє про успішне збереження.

Сторони:

Користувач, система.

Опис:

Користувач зберігає результат роботи у вибраному форматі.

Основний потік:

1. Користувач обирає «Зберегти».
2. Система пропонує формат:
    - MP3
    - OGG
    - FLAC
3. Користувач обирає шлях і параметри.
4. Система виконує експорт і кодування.
5. Показує повідомлення про успіх.

Винятки:

- Немає місця: система просить вибрати іншу теку.
- Неправильні параметри формату: показує рекомендації.
- Збій під час кодування: зберігає лог, пропонує повторити.

Користувач запускає аудіоредактор та може:

- Імпортувати аудіо
- Редагувати сегмент (копіювати, вставляти, вирізати, деформувати)
- Експортувати аудіо (OGG/FLAC/MP3)

- створити/відкрити/зберегти проєкт;

Адміністратор керує користувачами й налаштуваннями

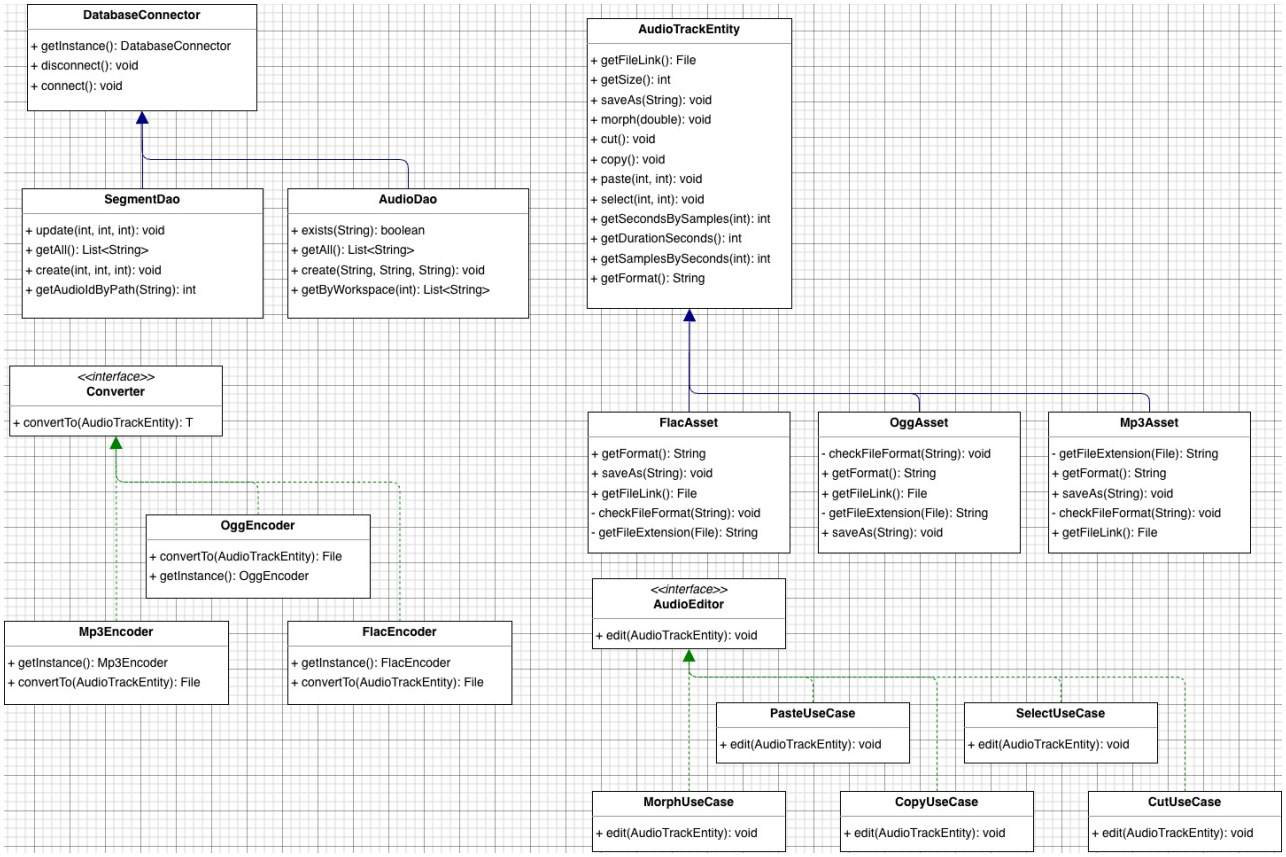

Рис 2. Діаграма класів

• DatabaseConnector: getInstance(), connect(), disconnect() - єдиний клас для роботи з підключенням до БД (Singleton, ізоляція доступу до БД).
• AudioDao: exists(String), getAll(), create(String, String, String), getByWorkspace(int) - DAO для аудіозаписів, працює через DatabaseConnector.
• SegmentDao: update(int, int, int), getAll(), create(int, int, int), getAudioIdByPath(String) - DAO для сегментів аудіо, також використовує DatabaseConnector.

Моделі (предметна область аудіодоріжки)

• AudioTrackEntity

- getFileLink(): File
- getSize(): int

- saveAs(String): void
- morph(double): void
- cut(): void
- copy(): void
- paste(int, int): void
- select(int, int): void
- getSecondsBySamples(int): int, getSamplesBySeconds(int): int
- getDurationSeconds(): int
- getFormat(): String

• FlacAsset

- getFormat(): String
- saveAs(String): void
- getFileLink(): File
- checkFileFormat(String): void
- getFileExtension(File): String

• OggAsset

- checkFileFormat(String): void
- getFormat(): String
- getFileLink(): File
- getFileExtension(File): String
- saveAs(String): void

• Mp3Asset

- getFileExtension(File): String

- getFormat(): String
- saveAs(String): void
- checkFileFormat(String): void
- getFileLink(): File

(FlacAsset, OggAsset, Mp3Asset - спеціалізації AudioTrackEntity для різних форматів файлів.)

Конвертери форматів (Encoder-и)
• Converter<T> (interface)

- convertTo(AudioTrackEntity): T

• OggEncoder

- convertTo(AudioTrackEntity): File
- getInstance(): OggEncoder

• Mp3Encoder

- getInstance(): Mp3Encoder
- convertTo(AudioTrackEntity): File

• FlacEncoder

- getInstance(): FlacEncoder
- convertTo(AudioTrackEntity): File

(Усі енкодери реалізують Converter<File> і використовують Singleton через getInstance().)

Редагування - use case шар
• AudioEditor (interface)

- edit(AudioTrackEntity): void

• PasteUseCase

- edit(AudioTrackEntity): void - реалізація вставки сегмента.

• MorphUseCase

- edit(AudioTrackEntity): void - реалізація деформації (стискання/розтягування).

• CopyUseCase

- edit(AudioTrackEntity): void - копіювання сегмента.

• CutUseCase

- edit(AudioTrackEntity): void - вирізання сегмента.

• SelectUseCase

- edit(AudioTrackEntity): void - вибір активної ділянки.

(Усі *UseCase* класи реалізують AudioEditor і інкапсулюють окремі операції редагування.)

Зв’язки між класами

• DatabaseConnector - використовується AudioDao та SegmentDao для роботи з БД.

• AudioDao, SegmentDao - забезпечують доступ до даних, не містять бізнес-логіки.

• FlacAsset, OggAsset, Mp3Asset - наслідуються від AudioTrackEntity та перевизначають поведінку для свого формату.

• Converter<T> - інтерфейс, який реалізують OggEncoder, Mp3Encoder, FlacEncoder.

• Encoder-и працюють з AudioTrackEntity як з джерелом даних і повертають вихідний File.

• AudioEditor - інтерфейс, який реалізують PasteUseCase, MorphUseCase, CopyUseCase, CutUseCase, SelectUseCase.

• UseCase-класи використовують методи AudioTrackEntity (cut, copy, paste, morph, select) для виконання конкретних дій редагування.

DatabaseConnector - це окремий клас, який відповідає за роботу з БД. Він ізолює логіку створення і закриття з’єднань від DAO-класів та решти бізнес-логіки.

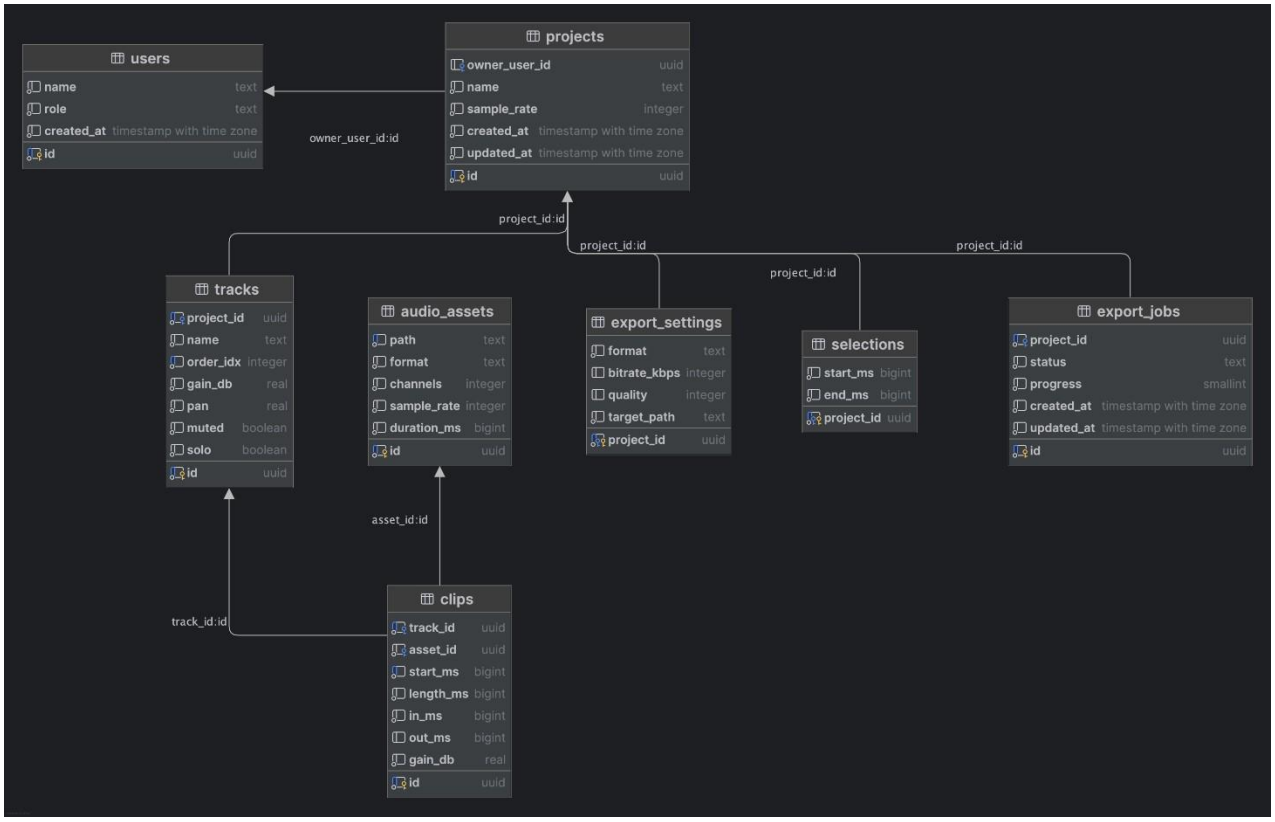

Рис 3. Структура бази даних

Висновок: У результаті виконання завдання було спроєктовано діаграму класів предметної області аудіоредактора. Діаграма відображає основні сутності системи: користувачів, проєкти, доріжки, кліпи, аудіоресурси, виділення, параметри експорту та історію експортів. Між класами визначено зв’язки та кардинальності, що дозволяє чітко зрозуміти структуру системи та взаємодію її компонентів. Розроблена модель є узгодженою з вимогами системи й може

слугувати основою для побудови структури бази даних та подальшої реалізації репозиторіїв відповідно до шаблону Repository.