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

Технології розроблення програмного забезпечення

Лабораторна робота №2

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

Виконала:
Студентка групи ІА-22
Микитенко Ірина

Перевірив:
Мягкий Михайло Юрійович

Київ 2024

Зміст

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

Мета: Проаналізувати тему, намалювати схему прецеденту, діаграму класів, розробити основні класи і структуру бази.

Хід роботи

1. Короткі теоретичні відомості

Діаграма варіантів використання

Мова UML – це універсальна мова для візуального моделювання, яка використовується для специфікації, візуалізації, проєктування та документування компонентів програмного забезпечення, бізнес-процесів та інших систем. UML дозволяє будувати концептуальні, логічні та графічні моделі складних систем, застосовуючи найкращі методи програмної інженерії.

В рамках UML моделі складних систем представлені у вигляді графічних конструкцій, званих діаграмами. Серед основних видів діаграм UML:

- діаграма варіантів використання (use case diagram),
- діаграма класів (class diagram),
- діаграма послідовностей (sequence diagram),
- діаграма станів (statechart diagram),
- діаграма діяльності (activity diagram),
- діаграма компонентів (component diagram),
- діаграма розгортання (deployment diagram).

Діаграма варіантів використання є початковою концептуальною моделлю системи. Вона відображає загальні вимоги до функціональності системи і не деталізує її внутрішню структуру.

Основні завдання діаграми варіантів використання:

1. Визначення меж функціональності системи.
2. Формулювання загальних вимог до функціональної поведінки.
3. Розробка початкової концептуальної моделі системи.
4. Створення основи для подальшого аналізу, проєктування та тестування.

Елементи діаграми:

- *Актор (actor)* – об'єкт або користувач, який взаємодіє із системою.

- *Варіант використання (use case)* – дії або послуги, які система надає актору.
- *Зв'язки (relationships)* – відношення між акторами та варіантами використання.

Діаграми UML. Діаграми класів. Концептуальна модель системи

Діаграми класів найчастіше використовуються при моделюванні програмних систем (ПС). Вони є формою статичного опису системи, показуючи її структуру, але не відображають динамічну поведінку об'єктів. На діаграмах класів зображуються класи, інтерфейси та відносини між ними.

Представлення класів

Клас – це основний будівельний блок ПС. Клас має назву, атрибути та операції. На діаграмі клас показується у вигляді прямокутника, розділеного на три області:

- Верхня область – назва класу.
- Середня область – опис атрибутів (властивостей).
- Нижня область – назви операцій (послуг), що надаються об'єктами цього класу.

Атрибути класу визначають структуру даних, які зберігаються в об'єктах. Кожен атрибут має ім'я та тип, що визначає його значення в програмі.

Видимість атрибутів

Для атрибутів класу можна задати видимість:

- *Відкритий (public)* – доступний для будь-якого класу.
- *Захищений (protected)* – доступний лише для нащадків.
- *Закритий (private)* – недоступний ззовні і використовується тільки в класі.

Це дозволяє реалізувати інкапсуляцію даних, забезпечуючи захист від несанкціонованого доступу.

Операції класу

Клас містить визначення операцій, які об'єкти цього класу повинні виконувати. Кожна операція має сигнатуру з іменем, типом повернення та списком параметрів. Закриті операції є внутрішніми для об'єктів класу, в той час як відкриті формують інтерфейс класу.

Відносини на діаграмах класів

На діаграмах класів зазвичай показуються асоціації та об'єднання (наследування):

1. *Асоціація (Association)* – відношення між об’єктами. Вона може мати назву та характеристику, таку як множинність, що показує, скільки об’єктів кожного класу може брати участь у зв’язку.
2. *Об'єднання (Generalization)* – показує зв'язок між класом-родителем та класом-нащадком. Цей зв'язок використовується для виявлення спільних характеристик кількох класів, які об’єднуються у батьківський клас.

Приклад застосування асоціацій

Асоціація "включає" показує, що один об'єкт може включати інші об'єкти. Наприклад, асоціація між класом "Супермаркет" і класом "Товар" показує, що супермаркет містить кілька товарів.

Асоціації можуть також мати свої атрибути та операції, у цьому випадку вони називаються клас-асоціацією.

Узагалі, асоціація є загальним видом зв’язку між класами, відображаючи використання одного класу іншим.

Логічна структура бази даних

Існує дві моделі баз даних: фізична та логічна. *Фізична модель* зберігає дані у вигляді бінарних файлів, оптимізованих для зберігання та отримання інформації. Логічна модель відображає структуру таблиць, представлень, індексів та інших елементів для програмування і використання бази даних.

Процес проєктування бази даних полягає в побудові зв'язків між програмними класами та таблицями. Основою для проєктування таблиць є нормальні форми, що допомагають уникнути надмірності та аномалій оновлення.

Нормальні форми:

1. 1НФ – кожен атрибут відношення має одне значення.
2. 2НФ – всі неключові атрибути залежать від ключа.
3. 3НФ – немає транзитивних залежностей між неключовими атрибутами.
4. НФ Бойса-Кодда – кожна функціональна залежність базується на ключі.

2. Тема

Тема 1. Музичний програвач

..1 Музичний програвач (iterator, command, memento, facade, visitor, client-server)

Музичний програвач становить собою програму для програвання музичних файлів або відтворення потокової музики з можливістю створення, запам'ятовування і редагування списків програвання, перемішування/повторення (shuffle/repeat), розпізнавання різних аудіо-форматів, еквалайзер.

3. Схема прецеденту

Схема прецеденту, що відповідає обраній темі, зображена на рисунку 1.

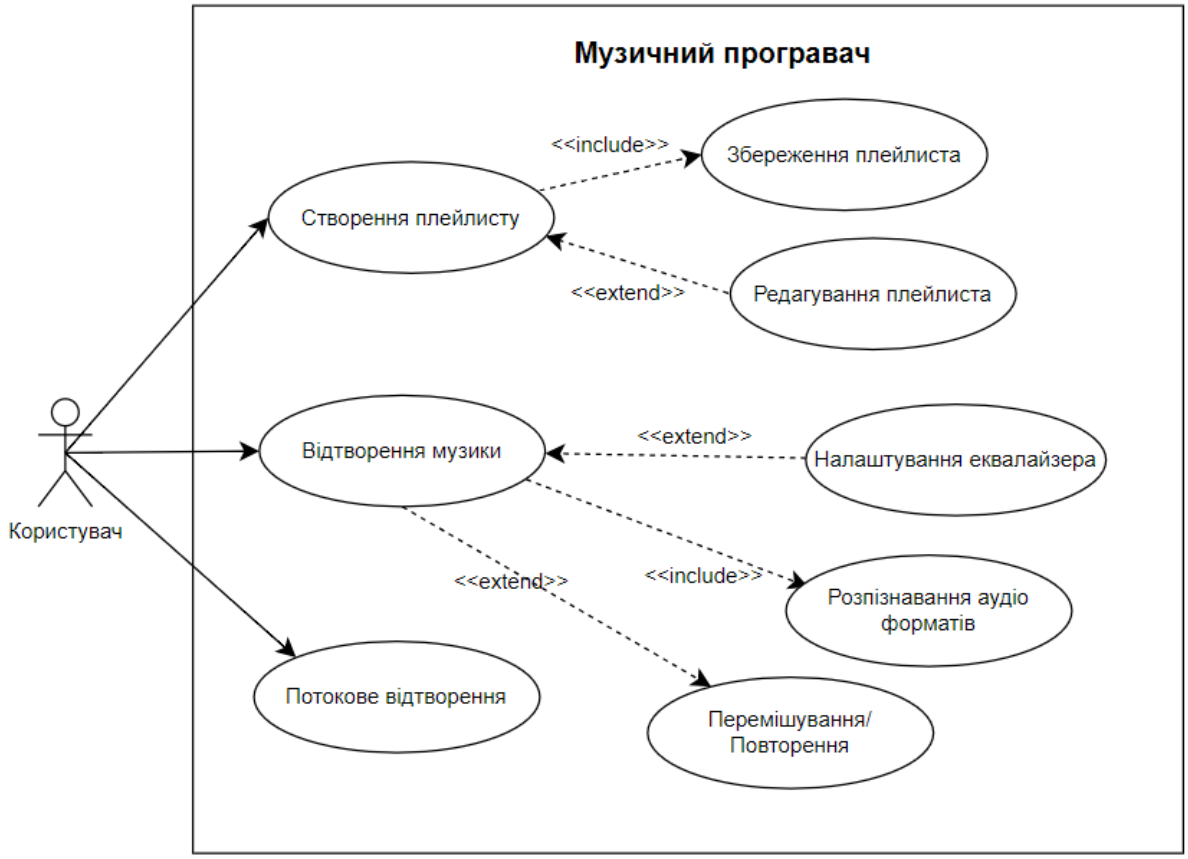

Рисунок 1

4. Сценарії використання для 3 прецедентів

2. Оберемо 3 прецеденти і напишемо для них сценарії використання

Прецедент: Програвати музику

Передумови: Користувач запустив музичний програвач.

Післяумови: Музичний файл відтворюється або процес відтворення зупиняється у разі виникнення помилки.

Актор: Користувач
Опис: Користувач запускає відтворення музичного файлу через програвач.

Основний сценарій:

1. Користувач відкриває музичний програвач.
2. Користувач обирає трек зі свого плейлиста або шукає музичний файл на пристрої.
3. Програма перевіряє аудіо формат і розпізнає його.
4. Програма починає відтворення музичного файлу.
5. Якщо потрібно, користувач може використовувати функції перемішування або повторення треків.
6. Користувач може налаштовувати еквалайзер для зміни якості звуку.
7. Після завершення треку система автоматично переходить до наступного або зупиняється, якщо увімкнено режим без повторення.

Винятки:

- Якщо аудіо файл пошкоджений або не підтримується, система відображає повідомлення про помилку і пропонує обрати інший файл.

Примітки:

- Користувач може налаштувати еквалайзер в будь-який момент під час відтворення.
- Відтворення можна зупинити або призупинити у будь-який момент за допомогою відповідних кнопок управління.

Прецедент: Створити список відтворення

Передумови:

1. Користувач запустив програму і має доступ до розділу з плейлистами.

2. Музичні файли доступні для додавання до плейлиста.

Післяумови: Створено новий плейлист з доданими треками, який доступний для відтворення.

Актор: Користувач

Опис: Користувач створює новий плейлист для зберігання улюблених треків.

Основний сценарій:

1. Користувач відкриває розділ зі списками відтворення.
2. Користувач натискає кнопку "Створити новий список".
3. Програма запитує назву для нового плейлиста.
4. Користувач вводить назву і підтверджує дію.
5. Програма створює новий плейлист і зберігає його.
6. Користувач додає треки до плейлиста.
7. Система підтверджує, що список збережено, та показує його в переліку плейлистів.

Винятки:

- Якщо користувач не вказав назву або залишив її порожньою, система виводить повідомлення про помилку і пропонує повторно ввести назву.
- Якщо користувач спробує додати трек, який вже є у списку, система повідомить про це та не додасть його вдруге.

Примітки:

- Користувач може редагувати плейлист після його створення, додаючи або видаляючи треки.

<u>Прецедент: Редагувати список відтворення</u>

Передумови: Користувач має доступ до існуючого плейлиста.

Післяумови: Зміни збережено, плейлист оновлений і доступний для подальшого використання.

Актор: Користувач

Опис: Користувач редагує вже існуючий плейлист.

Основний сценарій:

1. Користувач відкриває існуючий плейлист.
2. Користувач натискає кнопку "Редагувати".

3. Користувач додає або видаляє треки з плейлиста.
4. Користувач натискає кнопку "Зберегти зміни".
5. Програма зберігає внесені зміни.

Винятки: відсутні.

Примітки:

- Користувач може редагувати плейлист у будь-який момент, додаючи або видаляючи треки.
- Зміни будуть застосовані негайно після їх збереження і доступні для подальшого відтворення.

5. Діаграма класів

Діаграма класів зображена на рисунку 2.

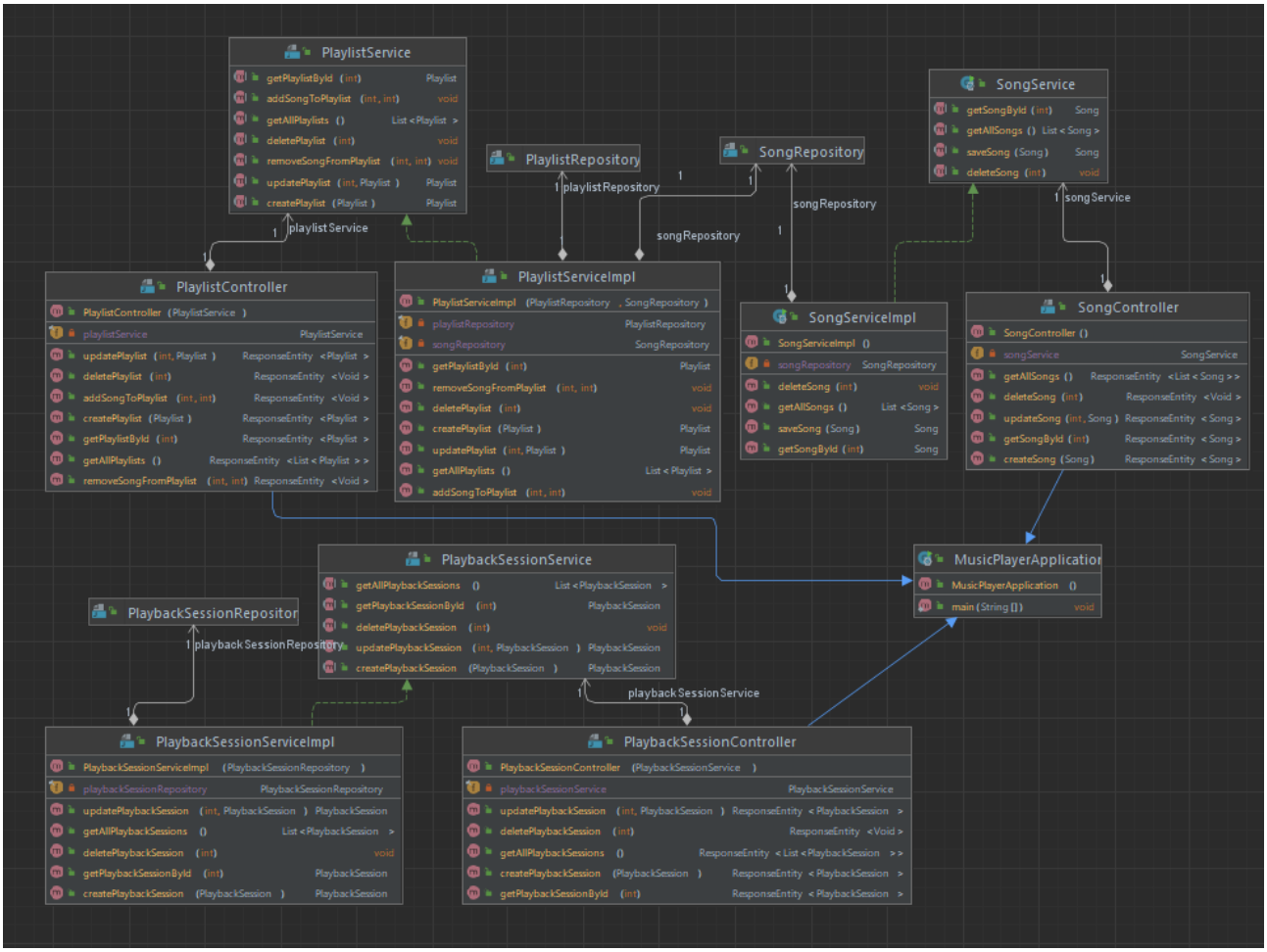

Рисунок 2

Шаблон репозиторію зображено на рисунку 3.

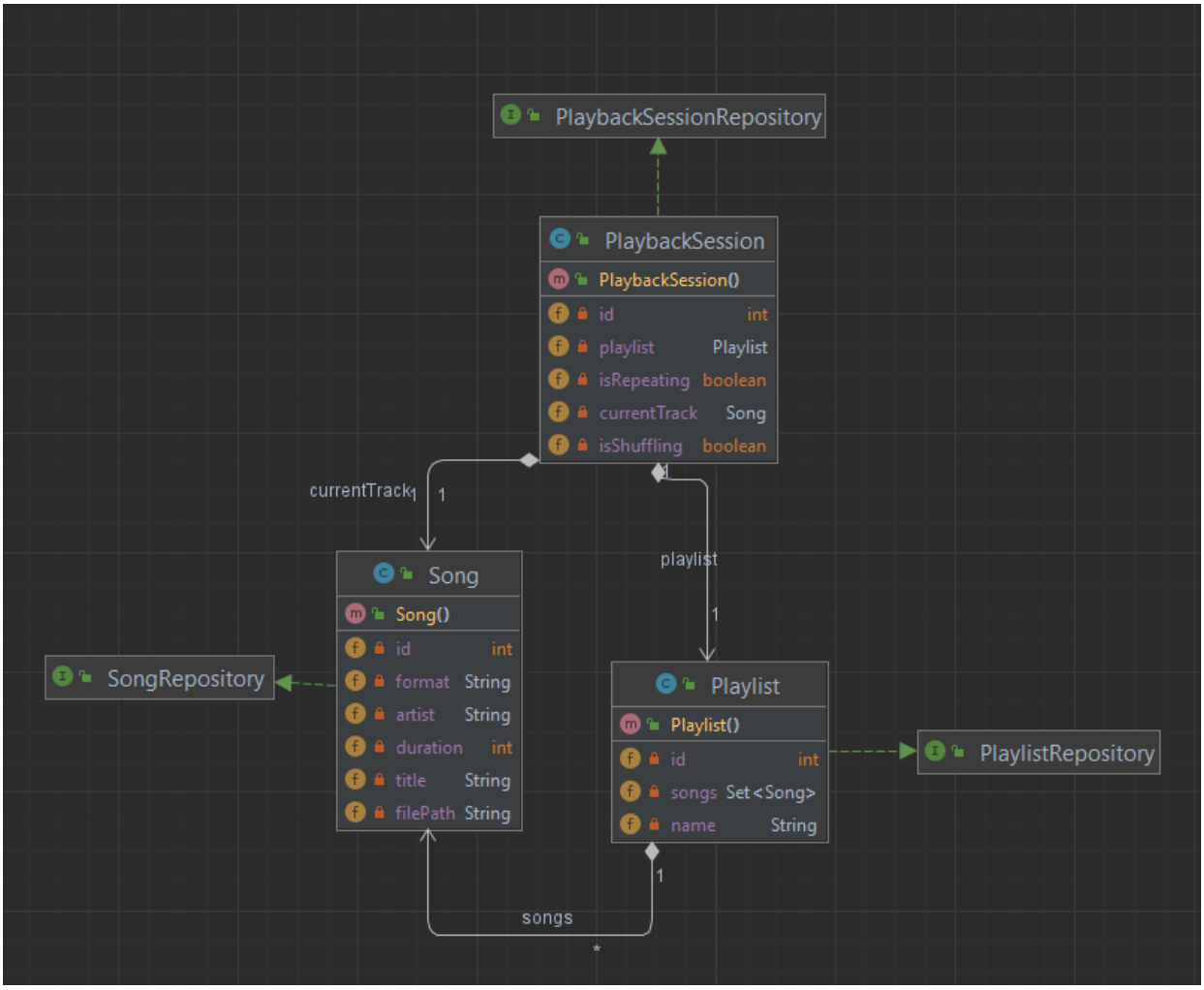

Рисунок 3

6. Структура бази даних

Структура бази даних зображена на Рисунку 4.

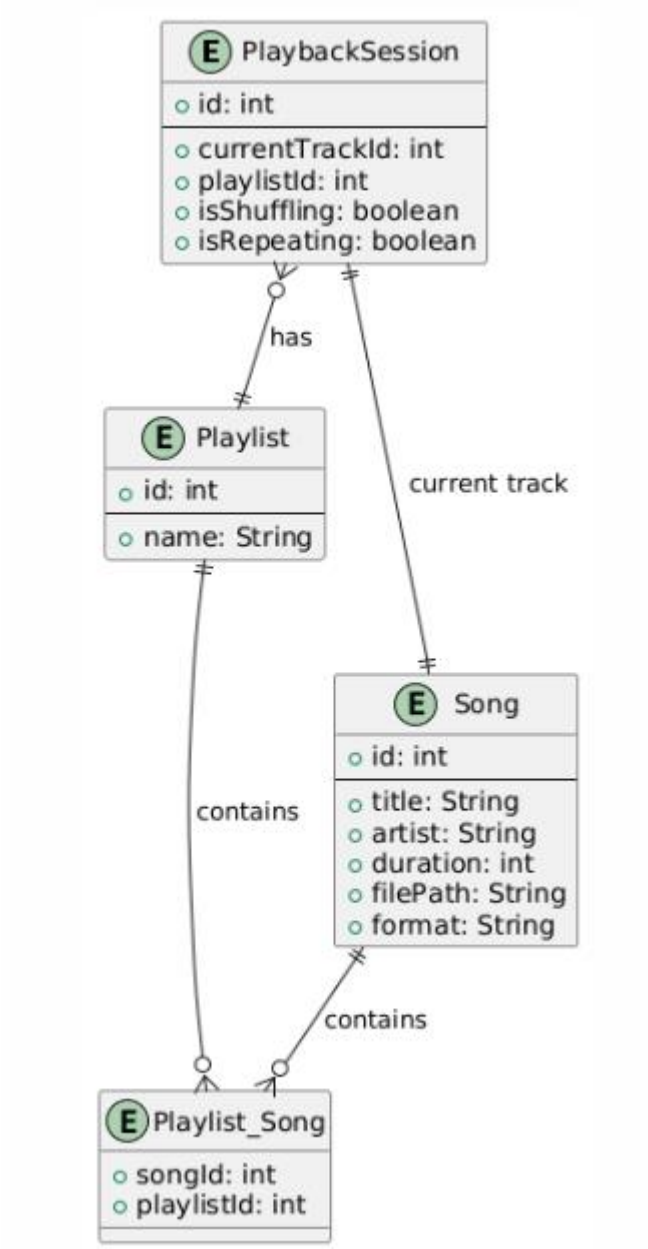

Рисунок 4

Висновок: в даній лабораторній роботі я ознайомилась з теоретичними матеріалом, проаналізувала тему, намалювала схему прецедентів, діаграму класів, розробила основні класи і структуру бази даних.