Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: «**СТВОРЕННЯ ЕЛЕКТРОННОЇ БІБЛІОТЕКИ ШАБЛОНІВ ПРОЕКТУВАННЯ**»

Виконав: студент 3-го року навчання,

Освітньої програми «Інженерія програмного забезпечення», 121

Дейнека Артем Валентинович

Керівник Бублик В.В.
доцент, кандидат фіз.-мат. наук

Київ – 2023

Зміст

Вступ

Програмні продукти сьогодні містять величезні об’єми коду. Цей код потрібно розробляти і підтримувати, причому вимоги до якості, швидкодії і швидкості розробки зросли і будуть зростати надалі. Потрібні уніфіковані рішення, які дозволяють швидко розробляти ефективний, і підтримуваний програмний код. Паттерни проектування є саме такими рішеннями. Вони зарекомендували себе як прості, надійні та швидкі способи вирішення типових проблем при розробці програмного забезпечення. Кожен розробник повинен знати їх щоб ефективно змінювати уже існуючий код і правильно проектувати новий. Про шаблони проектування було написано багато літератури, причому значна її частина знаходиться у відкритому доступі. Застосунок, який міг би структурувати усі наявні дані і надати можливість ефективно здійснювати пошук потрібної інформації може сильно спростити процес вивчення паттернів, а також спростить пошук необхідної інформації про них.

Розділ 1. Постановка задачі

Нашим завданням є створення застосунку, який дозволить переглядати інформацію про паттерни проектування та приклади програмного коду з їх використанням. Також має бути реалізований пошук паттернів та прикладів коду за:

- Назвою
- Типом паттерна
- Мовою програмування(тільки для вихідних кодів)
- Назвою проекту(тільки для вихідних кодів)

Також до мети нашої роботи входить розробка структури бібліотеки, яка забезпечить ефективний пошук за вищевказаними критеріями

Розділ 2. Особливості предметної області

2.1 Паттерни проектування, їх класифікація

Паттерни проектування - це загальноприйняті рішення для типових проблем, які можуть виникнути під час розробки програмного забезпечення. Вони представляють собою зразки архітектури або рішення, які можна використовувати для проектування програмних систем. Паттерни проектування допомагають створювати гнучкі, повторно використовувані та підтримувані програмні рішення.

Паттерни проектування можна класифікувати за різними критеріями. Один з найбільш відомих класифікаційних підходів - це поділ паттернів проектування на три категорії, вперше запропонований у книзі «Design Patterns: Elements of Reusable Object-Oriented Software»[1] також відомій як книга Банди Чотирьох:

1. Паттерни створення (Creational patterns): Ці паттерни стосуються процесу створення об'єктів. Вони допомагають зменшити залежність системи від типів об'єктів, які створюються, а також надають більш гнучкі способи створення об'єктів. До них належать Singleton (Одиночка), Factory Method (Фабричний метод) і Builder (Будівельник) та інші.
2. Паттерни структури (Structural patterns): Ці паттерни стосуються компонування об'єктів і класів в більш складні структури. Вони допомагають забезпечити гнучку взаємодію між об'єктами і розширення функціональності системи. Деякі популярні паттерни цієї категорії включають Adapter (Адаптер), Decorator (Декоратор) і Facade (Фасад).

[1] Gamma, E.; Helm, R.; Johnson, R. & Vlissides, J. M. (1994), *Design Patterns: Elements of Reusable Object-Oriented Software* , Addison-Wesley Professional .

3. Паттерни поведінки (Behavioral patterns): Ці паттерни стосуються взаємодії між об'єктами та розподілу обов'язків між ними. Вони допомагають організувати взаємодію між об'єктами більш гнучко та ефективно. Observer (Спостерігач), Strategy (Стратегія) і Template Method (Шаблонний метод) належать до цього типу паттернів.

Існують і інші способи класифікації, проте «поділ за призначенням» є найбільш відомим і впізнаваним, тому при створенні бібліотеки і при розділенні паттернів на типи ми використаємо саме поділ у нашій бібліотеці.

Розділ 3. Розробка структури репозиторію паттернів програмування

3.1 Загальна структура репозиторію

Інформація про паттерни так чи інакше повинна зберігатись у пам’яті комп’ютера. Ми розглянули два способи збереження цієї інформації: за допомогою засобів файлової системи і за допомогою баз даних. Останній спосіб має низку переваг:

- Швидкий і легкий пошук інформації при велиій кількості даних за рахунок кешування та індексів
- Ефективне представлення зв’язків між сутностями

Проте, наш репозиторій не міститиме велику кількість інформації про паттерни, а програма не повинна виконувати складні запити. Використання бази даних не надасть вищевказаних переваг, проте ускладнить реалізацію запитів і збільшить витрати пам’яті на СКБД. Тому нами було прийнято рішення зберігати інформацію про паттерни лише за допомогою засобів файлової системи.

3.2 Формат зберігання описів паттернів та прикладів програмного коду

3.2.1 Формат зберігання опису паттерна

Інформація про кожен паттерн включно із прикладами коду буде зберігатись в окремій для кожного з них директорії. Там буде міститись файл pattern_info.json який міститиме усю необхідну інформацію у форматі JSON. Цей файл повинен містити ім’я та тип паттерну, проте може містити додаткову інформацію. Такий спосіб зберігання дозволяє легко додавати нові типи паттернів, оскільки для цього достатньо вказати ім’я нового типу. Також це дозволяє реалізувати інші способи класифікації шаблонів проектування. Для зберігання опису шаблона було обрано формат PDF. Цей формат дозволяє зберігати ілюстрації(зокрема і діаграми класів) в одному файлі і підтримується багатьма платформами. Також

бібліотека Qt, яка використана при розробці застосунку містить засоби для роботи з цим форматом.

3.2.2 Формат зберігання прикладів коду

Приклади до коду зберігатимуться у папці code_examples, яка буде знаходитись у папці кожного із паттернів. Ця папка міститиме файл examples_info.json із іменами папок із проектами, у яких в свою чергу будуть містись файли вихідного коду та файл code_example_info.json із іменем проекту, мовою програмування на якій він написаний та списком файлів вихідного коду.

3.3 Формат зберігання даних про бібліотеку

Дані про бібліотеку будуть зберігатись у файлі book_info.json. Цей файл обов’язково повинен містити список папок із данними про паттерни, який описаний вище.

Розділ 4. Розробка застосунку для перегляду вмісту бібліотеки

4.1 Створення класів для репрезентації елементів бібліотеки

4.1.1 Клас Library

Для збереження інформації про всю бібліотеку було створено клас Library. Цей клас містить інформацію, зчитану з файлу book_info.json, а також список усіх паттернів, які зберігаються у цій бібліотеці за допомогою класу Pattern.

4.1.2 Клас Pattern

Даний клас зберігає інформацію про конкретний паттерн, а саме ім’я, тип, файл опису і об’єкт CodeExamples, який відповідає за доступ до прикладів програмного коду.

4.1.3 Клас CodeExamples

Даний клас зберігає приклади до коду, а також інформацю про них, зчитану з файлу examples_info.json. Конструктор цього класу забезпечує створення списку об’єктів класу Example, які відповідають за збереження інформації про приклад програмного коду

4.1.4 Клас Example

Клас відповідає за зберігання інформації про приклад програмного коду і зчитування файлів із вихідними кодами. Для досягнення останнього клас зберігає список об’єктів класу QFile, які мають потрібний API.

4.2 Розробка користувацького інтерфейсу

4.2.1 Загальний опис користувацького інтерфейсу

Користувацький інтерфейс складається із двох частин: списку паттернів/прикладів коду та частини, яка відображатиме обраний файл(див. Додаток). Частина, що містить список елементів бібліотеки представлена об’єктом класу QTabWidget, який у вкладці «Опис» містить рядок пошуку за іменем паттерну та список паттернів, представлений класом PatternTab, або схожа за дизайном, але представлена іншими класами вкладка «Приклади коду». Частина, що відповідає за відображення елементів – вказівник viewFrame типу QFrame. Він містить віджет, який відповідає за відображення обраного елемента

4.2.2 Опис нестандартних віджетів

Qt надає багато стандартних класів для реалізації тих чи інших елементів користувацького інтерфейсу. Проте, деякі з них потребують додаткової логіки, що і обумовило створення власних класів, що наслідують стандартні

4.2.2.1 ExampleTab

Клас наслідує QTreeView і використовується для відображення файлів вихідного коду прикладів до паттернів у вигляді дерева(див. Додаток). Клас також містить поля, потрібні для пошуку елементів даного дерева

4.2.2.2 PatternTab

Клас наслідує QTreeView і використовується для відображення списку паттернів для кожного із типів паттернів описаних у розділі 1. Клас також містить поля, потрібні для пошуку елементів даного дерева

4.2.2.3 SourceCodeView

Клас наслідує QTextEdit і використовується для відображення вихідного коду прикладів до паттернів. Клас також містить поля, потрібні для пошуку елементів даного дерева

4.2.2.4 PatternView

Клас наслідує QPdfView і використовується для відображення pdf документу з описом паттерна. Клас відповідає за коректне відображення опису паттерна.

4.2.2.5 TextEdit чи QPdfView

Елементи бібліотеки повинні відображатись як текстовий файл, або як пдф файл. Для цих задач у нас є два різних віджети, а тому неможливо вказати в якості типу елемента viewFrame один із вищевказаних класів. Це обумовило вибір типу QFrame як такого, що наслідується обома віджетами. При переключенні з однієї вкладки на іншу міняється віджет, який відповідає за відображення поточного елемента(див. Додаток).

4.3 Реалізація пошуку елементів бібліотеки за іменем

Для того, щоб реалізувати пошук паттернів за іменем клас PatternTab містить поле QHash<QTreeWidgetItem*, const Pattern*> _tabMap. Воно дозволяє для кожного елементу списку перевіряти чи відповідає відповідний йому паттерн заданій умові, а також зберігає вказівники на ті елементи, які були вилучені зі списку, щоб при потребі повернути назад. Аналогічна реалізація міститься у класі ExampleTab, проте там для кожного рівня вкладення дерева міститься свій словник де ключем є елемент, а значенням вказівник на батьківську вершину дерева. Пошук викликається за допомогою слоту search() в обох класах.

Висновок

У даній роботі було реалізовано можливість перегляду описів паттернів та прикладів їх використання. Також була реалізована можливість пошуку потрібної інформації

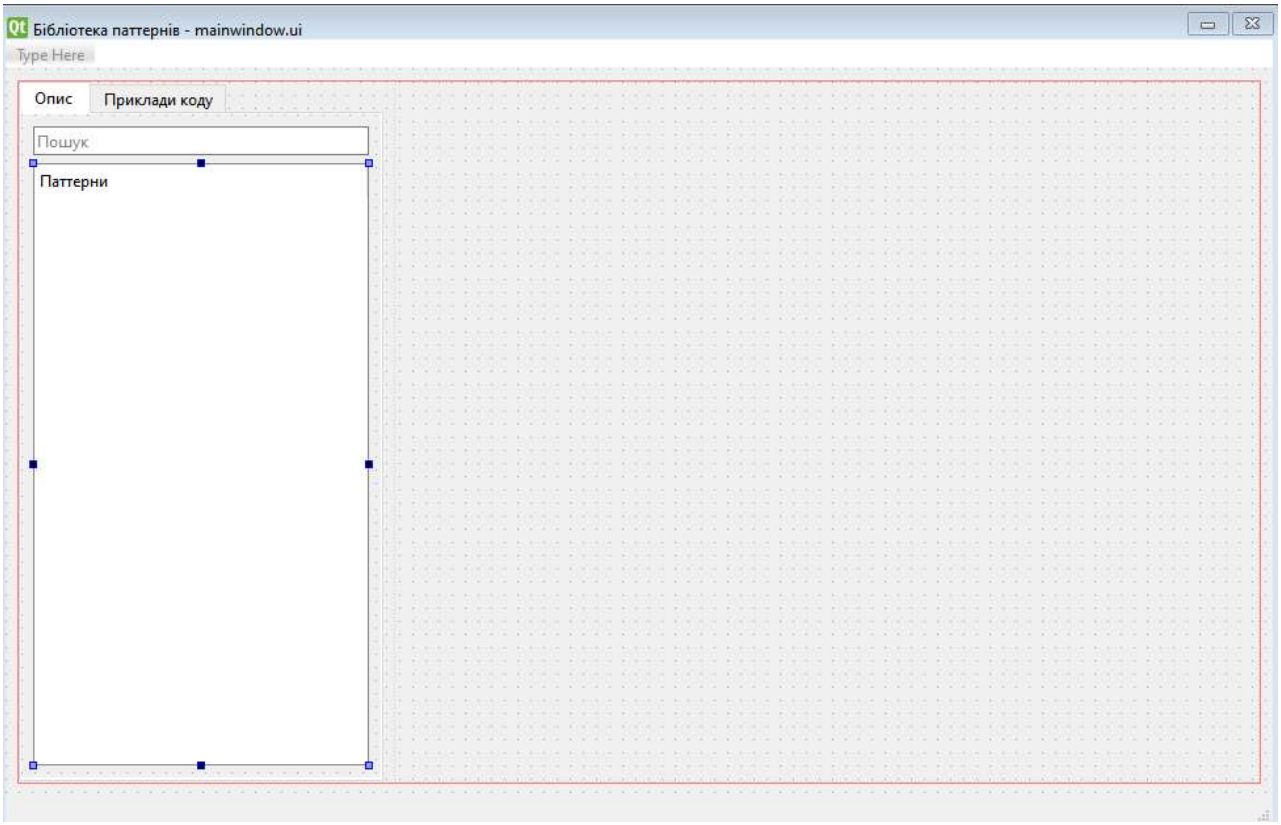
Застосунок можна покращити додавши можливість редагування вмісту бібілотеки(додавання нових паттернів та редагування вмісту існуючих елементів), пошук паттернів за ключовими словами у тексті опису.

Вагомим розширенням функціоналу може стати підтримка деяких поширених мов програмування(C/C++, Java, Python), а саме можливість запуску зібраних проектів.

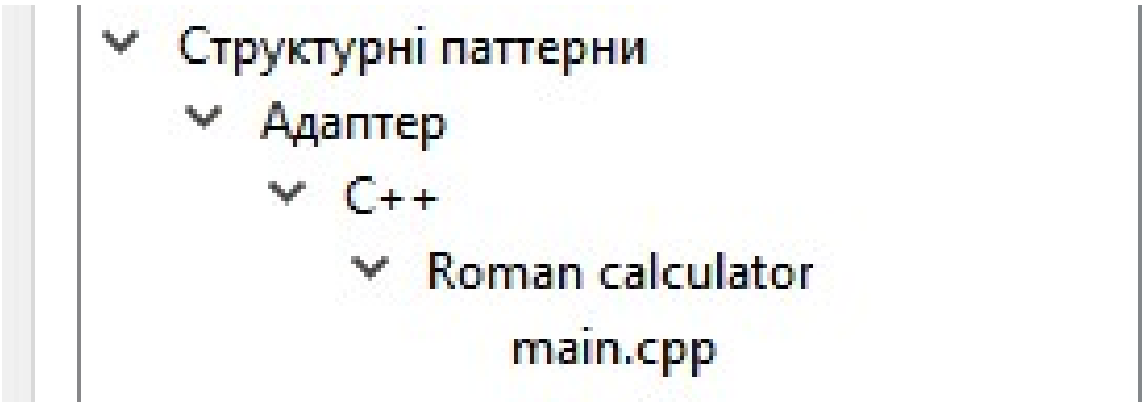
Джерела

1. Gamma, E.; Helm, R.; Johnson, R. & Vlissides, J. M. (1994), *Design Patterns: Elements of Reusable Object-Oriented Software* , Addison-Wesley Professional .

Додаток



Вигляд .ui файлу в qt designer



Приклад деревовидного представлення прикладів коду

```
void MainWindow::on_viewFrameChanged(QFrame* newFrame) {
    newFrame->setParent(this->ui->centralwidget);
    delete this->ui->viewFrame;
    this->ui->viewFrame = newFrame;
    this->ui->viewWidget->addWidget(newFrame);
}
```

Код, що відповідає за зміну фрейму, який відображає обраний елемент