

Створення електронної бібліотеки паттернів проектування

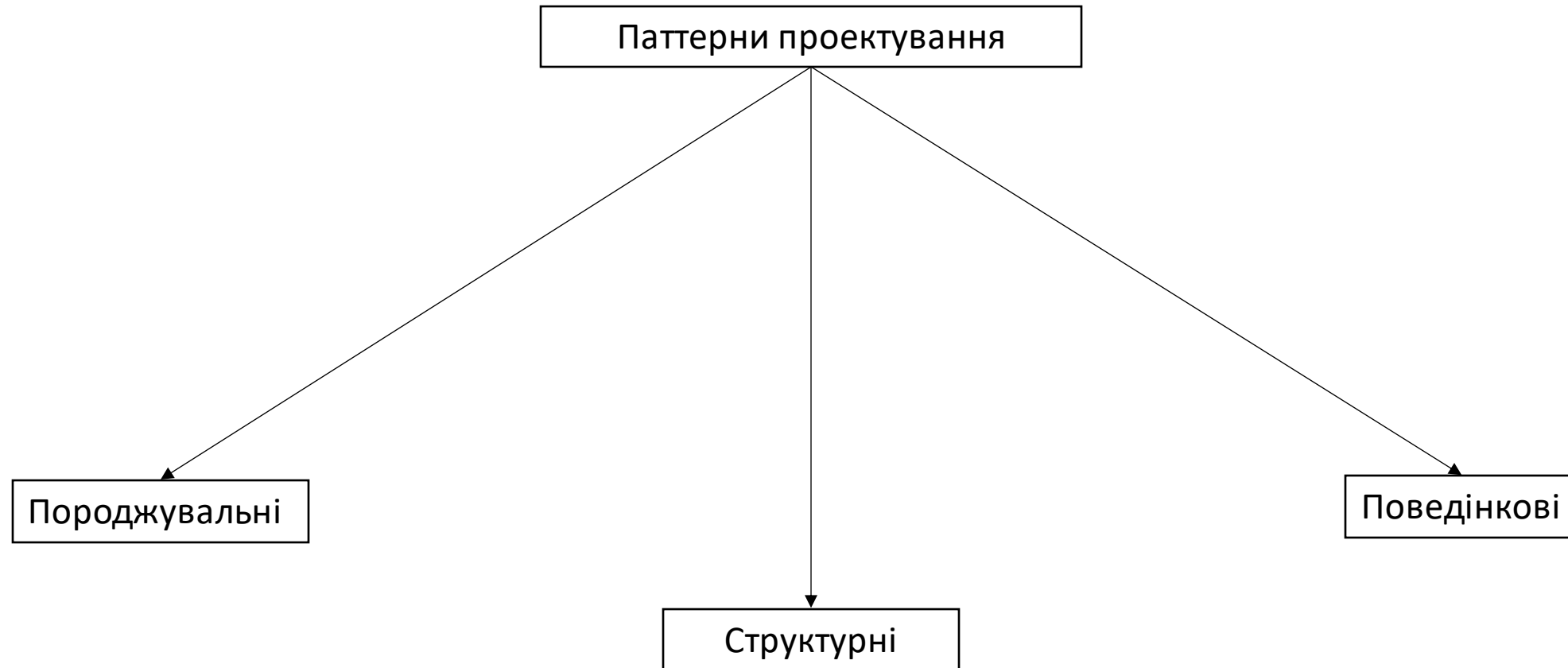
Виконав: студент 3-го року навчання,
Спеціальності: 121 Інженерія програмного забезпечення,
Дейнека Артем Валентинович

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

Постановка задачі

- Створення бібліотеки паттернів проектування із структурою, що дозволяє легко взаємодіяти з її наповненням
- Створення застосунку для перегляду вмісту бібліотеки
- Реалізація пошуку елементів бібліотеки за:
 - Назвою типу паттерну
 - Назвою паттерну
 - Назвою файлу вихідного коду
 - Назвою мови програмування, на якій написаний приклад програми

Чи єдиний це поділ?






Що використати для зберігання вмісту бібліотеки?



Зберігання паттерну

```
1  {  
2  "name": "Компонувальник",  
3  "pattern_type": "Структурні паттерни"  
4  }
```

Приклад вмісту файлу pattern_info.json

 code_examples	19/05/2023 9:02 pm	File folder
 description.pdf	19/05/2023 9:07 pm	Документ Adobe ...
 pattern_info.json	19/05/2023 9:07 pm	JSON File

Приклад вмісту папки паттерну

Зберігання прикладу програмного коду

```
{  
  "project_folders": ["example1"]  
}
```

Приклад списку папок з проектами

```
{  
  "name" : "New old rectangle",  
  "language": "C++",  
  "sources" : ["main.cpp"]  
}
```

Приклад вмісту файлу code_example_info.json

name	Date modified	type
code_example_info.json	19/05/2023 10:27 pm	JSON File
main.cpp	19/05/2023 10:23 pm	C++ Source File

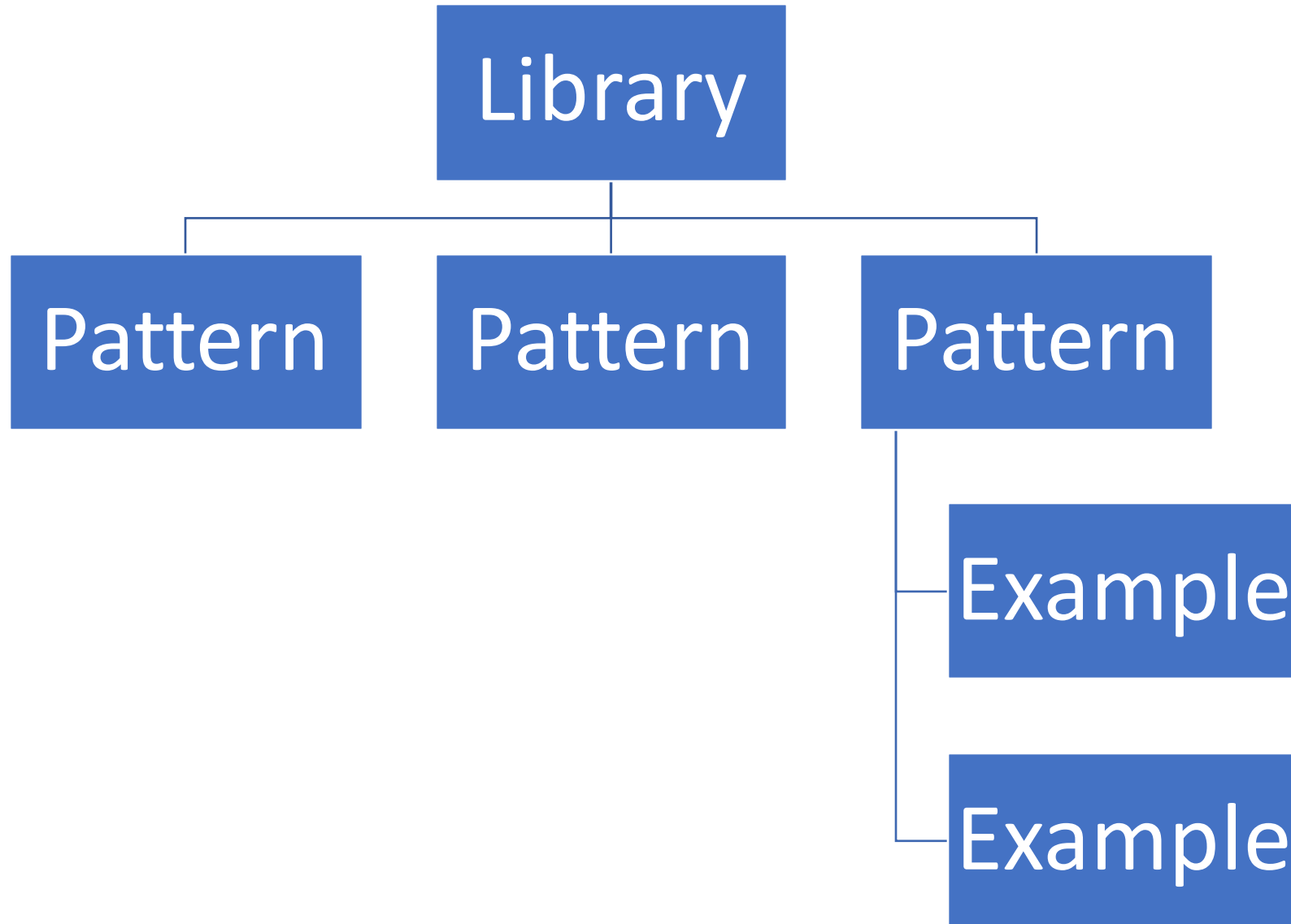
Приклад вмісту папки із проектом

Дані про бібліотеку

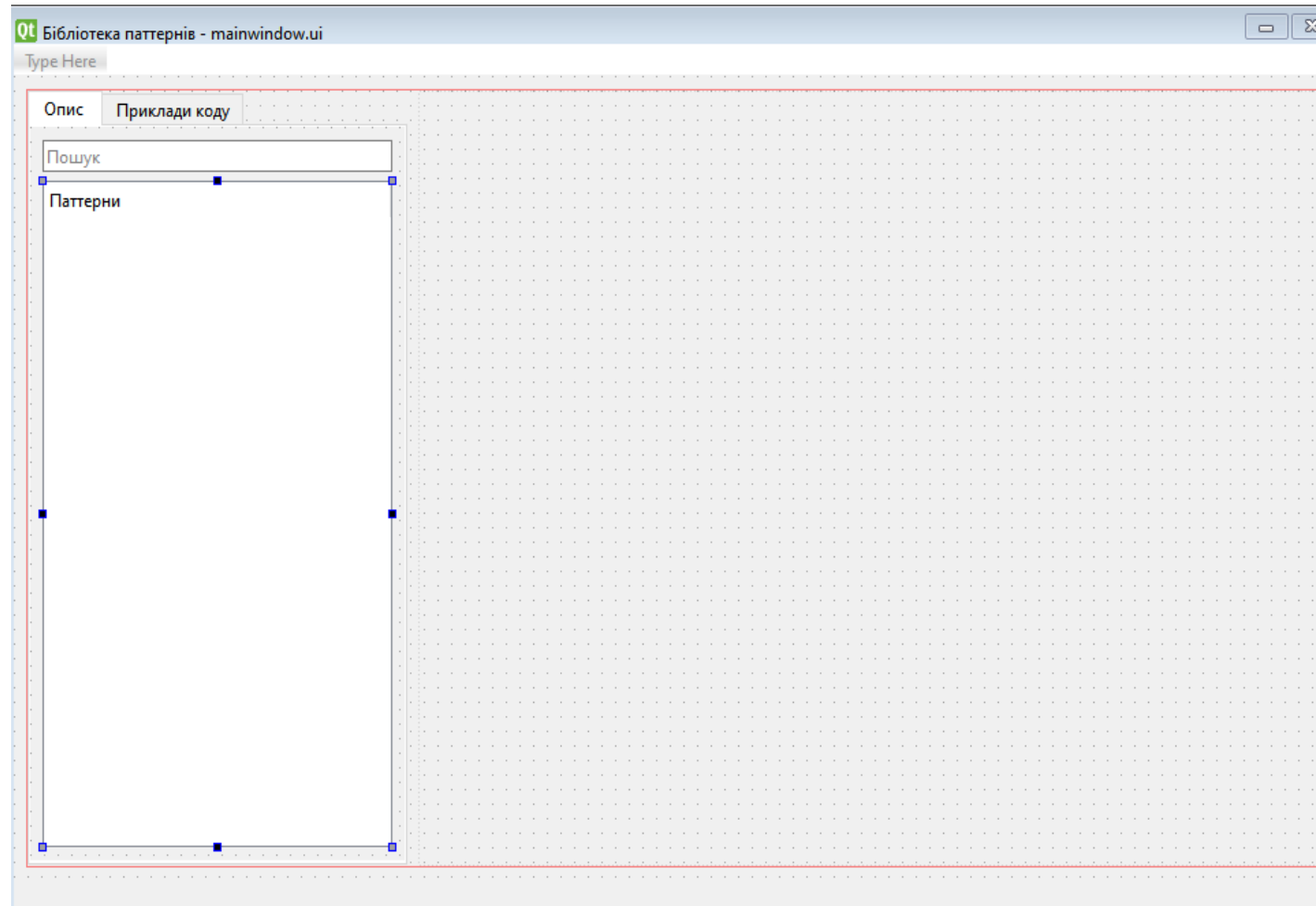
```
1  {
2  "folders": [
3      "adapter",
4      "visitor",
5      "strategy",
6      "template_method",
7      "singleton",
8      "observer",
9      "flyweight",
10     "iterator",
11     "mediator",
12     "memento",
13     "prototype",
14     "proxy",
15     "responsibility_chain",
16     "state",
17     "factory_method",
18     "abstract_factory",
19     "builder",
20     "bridge",
21     "command",
22     "composite",
23     "decorator",
24     "facade"
25 ]
26 }
```

Вміст файлу із списком папок, що містять дані про паттерни

Структура бібліотеки



Користувацький інтерфейс

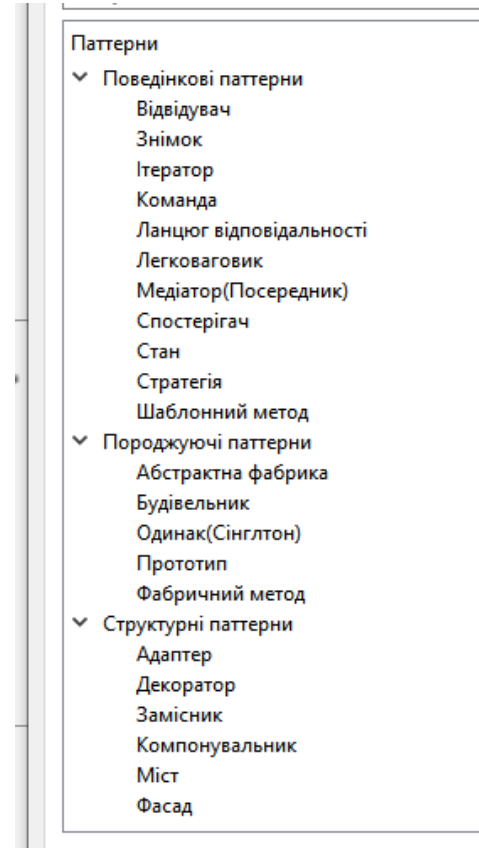


Вигляд .ui файлу у Qt Creator

Відображення елементів бібліотеки

```
✓ Породжуючі паттерни
✓ Структурні паттерни
  ✓ Адаптер
    ✓ C++
      ✓ New old rectangle
        main.cpp
```

Деревовидна структура
відображення файлів вихідного
коду



Список паттернів

Відображення елементів бібліотеки

The screenshot shows the 'Patterns Library' application with the 'Iterator' pattern selected in the left sidebar. The main content area displays the 'Проблема' (Problem) section for the Iterator pattern. It includes a description of the problem, a code example showing a simple array iteration, and a discussion of how the Iterator pattern solves the problem by abstracting the iteration logic.

```
const unsigned SIZE = 5;
int* arr = new int[SIZE];
for (int i = 0; i < SIZE; ++i)
    cout << arr[i] << endl;

LinkedList<int> list(arr, SIZE);
auto node = list.head; // порушення інкапсуляції!
while (node != nullptr) {
    cout << node->data << endl;
    node = node->next;
}
```

The screenshot shows the 'Patterns Library' application with the 'New old rectangle' pattern selected in the left sidebar. The main content area displays the 'Приклади коду' (Code Examples) section for the New old rectangle pattern. It includes a code example showing the implementation of the pattern in C++.

```
#include <iostream>
using namespace std;
typedef int Coordinate;
typedef int Dimension;

// Desired interface
class Rectangle {
public:
    virtual void draw() = 0;
};

// Legacy component
class LegacyRectangle
{
public:
    LegacyRectangle(Coordinate x1, Coordinate y1, Coordinate x2, Coordinate y2)
    {
        x1_ = x1;
        y1_ = y1;
        x2_ = x2;
        y2_ = y2;
        cout << "LegacyRectangle: create. (" << x1_ << ", " << y1_ << ") => ("
            << x2_ << ", " << y2_ << ")" << endl;
    }
    void oldDraw()
    {
        cout << "LegacyRectangle: oldDraw. (" << x1_ << ", " << y1_ <<
            ") => (" << x2_ << ", " << y2_ << ")" << endl;
    }
private:
    Coordinate x1_;
    Coordinate y1_;
    Coordinate x2_;
    Coordinate y2_;
};
```

Відображення опису та прикладу коду паттерну проектування

Пошук паттернів

Поля класу PatternTab, які призначені для пошуку

```
QHash<QTreeWidgetItem*, const Pattern*> _tabMap;  
QHash<QString, QTreeWidgetItem*> _nameMap;
```

Реалізація пошуку

```
void PatternTab::search(const QString& prefix) {  
    const Pattern* p = nullptr;  
    QTreeWidgetItem* type = nullptr;  
    for (QTreeWidgetItem* item : this->_tabMap.keys()) {  
        p = this->_tabMap[item];  
        type = this->_nameMap[p->type()];  
        if (p->name().startsWith(prefix)) {  
            type->addChild(item);  
        }  
        else {  
            type->removeChild(item);  
        }  
        type->sortChildren(0, Qt::AscendingOrder);  
    }  
}
```

Пошук паттернів

```
QHash<QString, QTreeWidgetItem*> _typeItemMap;  
QHash<QTreeWidgetItem*, QTreeWidgetItem*> _typePatternsMap;  
QHash<QTreeWidgetItem*, QTreeWidgetItem*> _patternLanguagesMap;  
QHash<QTreeWidgetItem*, QTreeWidgetItem*> _languageProjectsMap;  
QHash<QTreeWidgetItem*, QTreeWidgetItem*> _projectSourcesMap;
```

Поля класу ExampleTab, які призначені для пошуку

Висновки

- Було розроблено структуру бібліотеки паттернів проектування
- Створено застосунок для перегляду вмісту бібліотеки
- Реалізовано усі види пошуку, які необхідні за умовою задачі

Дякую за увагу