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

Лабораторна робота №8

ШАБЛОНИ «COMPOSITE»,
«FLYWEIGHT», «INTERPRETER»,
«VISITOR»

Варіант 10

Виконала
студентка групи ІА – 13:
Луценко Юлія

Перевірив:
Мягкий Михайло

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.

2. Реалізувати частину функціоналу робочої програми у вигляді класів і їх взаємодій для

досягнення конкретних функціональних можливостей.

3. Застосування одного з розглянутих шаблонів при реалізації програми.

Варіант:

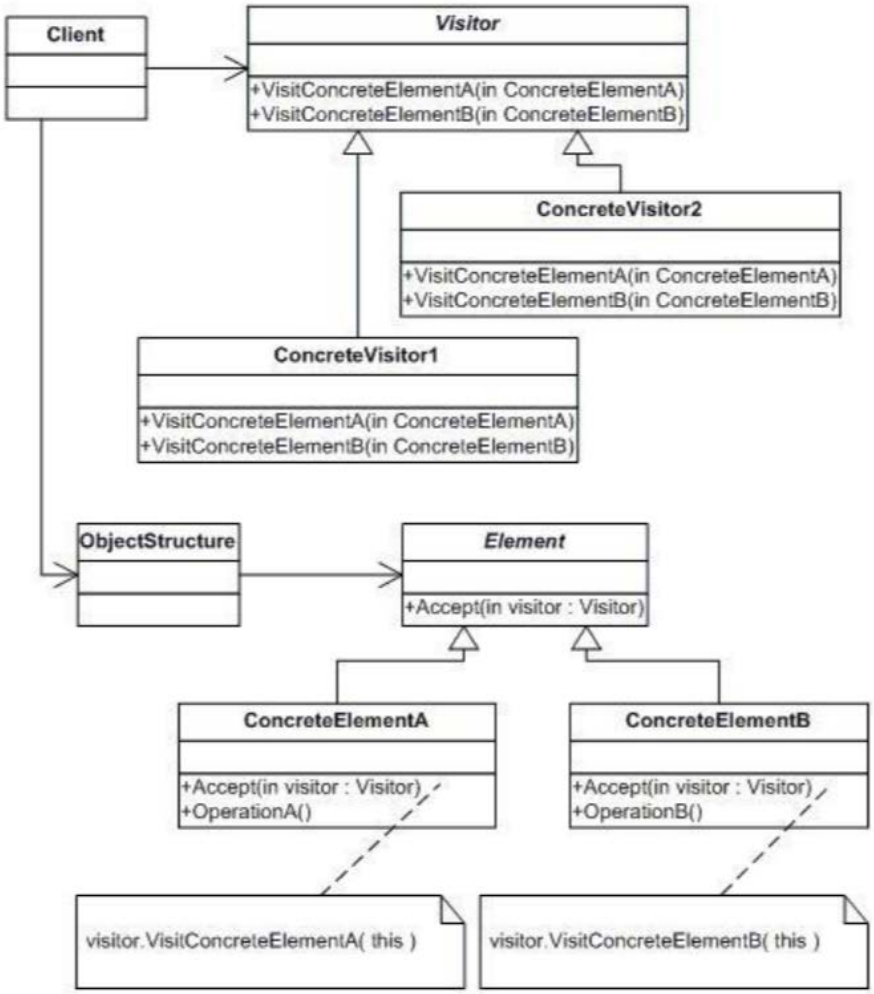
10. VCS all-in-one (iterator, adapter, factory method, facade, visitor, p2p)

Клієнт для всіх систем контролю версій повинен підтримувати основні команди і дії (commit, update, push, pull, fetch, list, log, patch, branch, merge, tag) для 3-х основних систем управління версіями (svn, git, mercurial), а також мати можливість вести реєстр репозиторіїв (і їх типів) і відображати дерева фіксації графічно

Хід роботи

Паттерн Відвідувач(Visitor)

Відвідувач — це поведінковий патерн проектування, що дає змогу додавати до програми нові операції, не змінюючи класи об’єктів, над якими ці операції можуть виконуватися.



Інтерфейс VCSVisitor

```python
from abc import ABC, abstractmethod


2 usages  new *
class VCSVisitor(ABC):
    1 usage (1 dynamic)  new *
    @abstractmethod
    def visit_git(self, git_vc):
        pass

    1 usage (1 dynamic)  new *
    @abstractmethod
    def visit_mercurial(self, mercurial_vc):
        pass

    1 usage (1 dynamic)  new *
    @abstractmethod
    def visit_svn(self, svn_vc):
        pass
```

MyVCSVisitor(клас,що має методи для відвідування різних сисем контролю версій)

```
class MyVCSVisitor(VCSVisitor):
    1 usage (1 dynamic)  new *
    def visit_git(self, git_vc):
        git_vc.show_repositories()

    1 usage (1 dynamic)  new *
    def visit_mercurial(self, mercurial_vc):
        mercurial_vc.show_repositories()

    1 usage (1 dynamic)  new *
    def visit_svn(self, svn_vc):
        svn_vc.show_repositories()
```

Кожен клас систему контролю версій має метод accept(),що приймає візітор та «відвідує»

```
def accept(self, visitor):
    visitor.visit_git(self)
```

```
def accept(self, visitor):
    visitor.visit_svn(self)
```

```
def accept(self, visitor):
    visitor.visit_mercurial(self)
```

Використання візітора в класі Facade

```
def show_repositories(self, version_control):
    version_control.accept(MyVCSVisitor())
```

Паттерн Visitor дозволяє визначити нові операції над об'єктами, не змінюючи їхнього класу. У нашому випадку, операції визначаються методами в класі myVCSVisitor, який є імплементує інтерфейс VCSVisitor.

Це дозволяє динамічно визначати, який метод потрібно викликати залежно від команди. Паттерн Visitor особливо корисний, коли є багато різних операцій, які можуть бути виконані над об'єктами, і потрібно дотримуватися принципу open/closed principle - можливість додавати нові операції без зміни існуючого коду.

Висновок: я реалізувала патерн visitor,що дозволяє динамічно визначати, який метод потрібно викликати залежно від команди.