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 №4
ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»

Варіант 10

Виконала
студентка групи ІА – 13:
Луценко Юлія Сергіївна

Перевірив:
М’який Михайло

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.

2. Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії

для досягнення конкретних функціональних можливостей.

3. Застосування одного з розглянутих шаблонів при реалізації програми

Варіант:

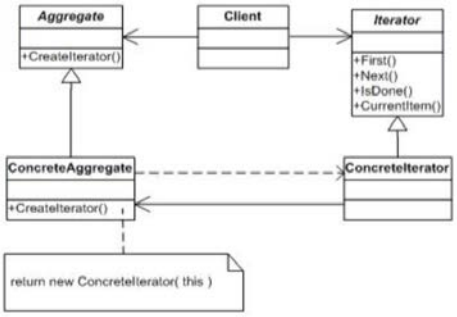
10. VCS all-in-one (iterator, adapter, factory method, facade, visitor, p2p)
Клієнт для всіх систем контролю версій повинен підтримувати основні команди і дії (commit, update, push, pull, fetch, list, log, patch, branch, merge, tag) для 3-х основних систем управління версіями (svn, git, mercurial), а також мати можливість вести реєстр репозиторіїв (і їх типів) і відображати дерева фіксації графічно

Хід роботи

Шаблон Ітератор

«Iterator» (Ітератор) являє собою шаблон реалізації об'єкта доступу до набору (колекції, агрегату) елементів без розкриття внутрішніх механізмів реалізації. Ітератор виносить функціональність перебору колекції елементів з самої колекції, таким чином досягається розподіл обов'язків: колекція відповідає за зберігання даних, ітератор - за прохід по колекції.

Шаблон «Iterator»



Ідея патерна Ітератор полягає в тому, щоб винести поведінку обходу колекції з самої колекції в окремий клас

Код

Інтерфейс ітератора:

```
class RepositoryIteratorInterface(ABC):
    yuliiaaaaa
    @abstractmethod
    def first(self):
        pass

    yuliiaaaaa
    @abstractmethod
    def next(self):
        pass

    yuliiaaaaa
    @abstractmethod
    def isDone(self):
        pass

    yuliiaaaaa
    @abstractmethod
    def currentItem(self):
        pass
```

Реалізація для Гіта

```python
class GitRepositoryIterator(RepositoryIteratorInterface):
    yuliiaaaaa
    def __init__(self, repositories: List[RepositoryDTO]):
        self.repositories = [repo for repo in repositories if repo.vcs_type == "Git"]
        self.index = 0


    yuliiaaaaa
    def __iter__(self):
        return self

    yuliiaaaaa
    def __next__(self):
        if self.index >= len(self.repositories):
            raise StopIteration
        repo = self.repositories[self.index]
        self.index += 1
        return repo

    yuliiaaaaa *
    def first(self):
        if len(self.repositories) > 0:
            self.index = 0
        else:
            raise ValueError("Collection is empty")
```

```python
def next(self):
    if self.isDone():
        raise StopIteration("Reached end of the collection")
    repo = self.repositories[self.index]
    self.index += 1
    return repo

2 usages  yuliiaaaaa
def isDone(self):
    return self.index >= len(self.repositories)

yuliiaaaaa
def currentItem(self):
    if self.isDone():
        raise ValueError("Iterator is at the end, no current item available")
    return self.repositories[self.index]
```

Реалізація для SVN

```python
class SVNRepositoryIterator(RepositoryIteratorInterface):
    yuliiaaaaa
    def __init__(self, repositories: List[RepositoryDTO]):
        self.repositories = [repo for repo in repositories if repo.vcs_type == "SVN"]
        self.index = 0

    yuliiaaaaa
    def __iter__(self):
        return self

    yuliiaaaaa
    def __next__(self) -> RepositoryDTO:
        if self.isDone():
            raise StopIteration("Reached end of the collection")
        repo = self.repositories[self.index]
        self.index += 1
        return repo

    yuliiaaaaa *
    def first(self):
        if len(self.repositories) > 0:
            self.index = 0
        else:
            raise ValueError("Collection is empty")
```

```python
def next(self) -> RepositoryDTO:
    if self.isDone():
        raise StopIteration("Reached end of the collection")
    repo = self.repositories[self.index]
    self.index += 1
    return repo

3 usages  yuliiaaaaa
def isDone(self) -> bool:
    return self.index >= len(self.repositories)

yuliiaaaaa
def currentItem(self) -> RepositoryDTO:
    if self.isDone():
        raise ValueError("Iterator is at the end, no current item available")
    return self.repositories[self.index]
```

Реалізація для Mercurial

```python
class MercurialRepositoryIterator(RepositoryIteratorInterface):
    yuliiaaaaa
    def __init__(self, repositories: List[RepositoryDTO]):
        self.repositories = [repo for repo in repositories if repo.vcs_type == "Mercurial"]
        self.index = 0

    yuliiaaaaa
    def __iter__(self):
        return self

    yuliiaaaaa
    def __next__(self) -> RepositoryDTO:
        if self.isDone():
            raise StopIteration("Reached end of the collection")
        repo = self.repositories[self.index]
        self.index += 1
        return repo

    yuliiaaaaa *
    def first(self):
        if len(self.repositories) > 0:
            self.index = 0
        else:
            raise ValueError("Collection is empty")
```

```python
def next(self) -> RepositoryDTO:
    if self.isDone():
        raise StopIteration("Reached end of the collection")
    repo = self.repositories[self.index]
    self.index += 1
    return repo

3 usages  yuliiaaaaa
def isDone(self) -> bool:
    return self.index >= len(self.repositories)

yuliiaaaaa
def currentItem(self) -> RepositoryDTO:
    if self.isDone():
        raise ValueError("Iterator is at the end, no current item available")
    return self.repositories[self.index]
```

Виклик у main(обирається певний ітератор залежно від vcs)

```
def show_repositories_for_vcs(vcs_type):
    repositories = Repository(connection).find_all()
    if vcs_type == "Git":
        iterator = GitRepositoryIterator(repositories)
    elif vcs_type == "Mercurial":
        iterator = MercurialRepositoryIterator(repositories)
    elif vcs_type == "SVN":
        iterator = SVNRepositoryIterator(repositories)
    else:
        print("Invalid VCS type")
        return

    # Display repositories for the chosen VCS type
    print(f"\nRepositories for {vcs_type}:")
    for repository in iterator:
        print(Fore.BLUE + f"Name: {repository.name}, "
      f"VCS Type: {repository.vcs_type}, URL: {repository.url}"+Fore.GREEN)
```

Результат для vcs Mercurial

```
Repositories for Mercurial:
Name: myRepo, VCS Type: Mercurial, URL: D:\ТПР3\myRepo
Name: repoMer, VCS Type: Mercurial, URL: D:\ТПР3\repoMer
Selected Version Control System: Mercurial
Current Repository: D:\ТПР3\myRepo
Choose an action:
```

Результат для vcs SVN

```
Repositories for Git:
Name: repo, VCS Type: Git, URL: D:\ТПР3\repo
Selected Version Control System: Git
Current Repository: D:\ТПР3\repo
Choose an action:
1. Commit
2. Watch History
3. Initialize Repository
```

Висновок:у цій лабораторній було реалізовано патерн ітератор,що дало змогу нам проходитись по даним наших репозиторіїв.Були реалізовані різні ітератори для різних систем версій,що дало нам змогу проходитись по репозиторіям різними ітераторами окремо,залежно від самого типу системи контролю версій.