МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ В.Н. КАРАЗІНА
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК

Лабораторна робота №6

з дисципліни «ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ»

Тема: «Шаблони проектування»

Виконала:
студентка групи КС-33
Рузудженк С.Р.
Перевірив:
ст. викл. Товстокоренко О.Ю.

Харків – 2019

Лабораторна робота №6

Тема: Шаблони проектування

Мета: ознайомитися із патернами проектування та їх реалізацією

Хід роботи

У ході виконання даної лабораторної роботи було проведено ознайомлення з патернами проектування та розроблена програмна реалізація деяких з них.

Патерн проектування — це типовий спосіб вирішення певної проблеми, що часто зустрічається при проектуванні архітектури програм.

На відміну від готових функцій чи бібліотек, патерн не можна просто взяти й скопіювати в програму. Патерн являє собою не якийсь конкретний код, а загальний принцип вирішення певної проблеми, який майже завжди треба підлаштовувати для потреб тієї чи іншої програми.

Описи патернів зазвичай дуже формальні й найчастіше складаються з таких пунктів:

- проблема, яку вирішує патерн;
- мотивація щодо вирішення проблеми способом, який пропонує патерн;
- структура класів, складових рішення;
- приклад однією з мов програмування;
- особливості реалізації в різних контекстах;
- зв’язки з іншими патернами.

У даній лабораторній роботі проведено детальне ознайомлення з двома поведінковими шаблонами проектування: Observer (Спостерігач) та Memento (Зберігач) .

1. Спостерігач

Даний патерн дозволяє об’єктам спостерігати за діями, що виконуються в інших об’єктах.

Програмна реалізація:

Створимо клас Університет, клас Керуючий та клас для тестування методів, а також безпосередньо інтерфейс, що реалізує патерн проектування.

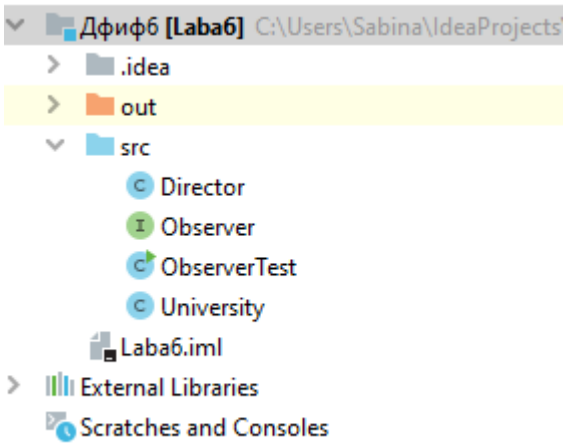

Рисунок 1 – Структура проекту

Далі наведено лістинг коду (ПР шаблону проектування Спостерігач)

```
import java.util.ArrayList;
import java.util.List;
interface Observer {
    void event(List<String> strings);
}
class University {
    private List<Observer> observers = new ArrayList<Observer>();
    private List<String> students = new ArrayList<String>();
    public void addStudent(String name) {
        students.add(name);
        notifyObservers();
    }
    public void removeStudent(String name) {
        students.remove(name);
        notifyObservers();
    }
    public void addObserver(Observer observer){
        observers.add(observer);
    }
    public void removeObserver(Observer observer) {
        observers.remove(observer);
    }
    public void notifyObservers(){
        for (Observer observer : observers) {
            observer.event(students);
        }
    }
}
class Director implements Observer {
    public void event(List<String> strings) {
        System.out.println("The list of students has changed: " + strings);
```

```
    }
}

public class ObserverTest {//тест
    public static void main(String[] args) {
        University university = new University();
        Director director = new Director();
        university.addStudent("Vaska");
        university.addObserver(director);
        university.addStudent("Anna");
        university.removeStudent("Vaska");
    }
}
```

Для перевірки роботи програми необхідно зкомпілювати та запустити тестовий клас.

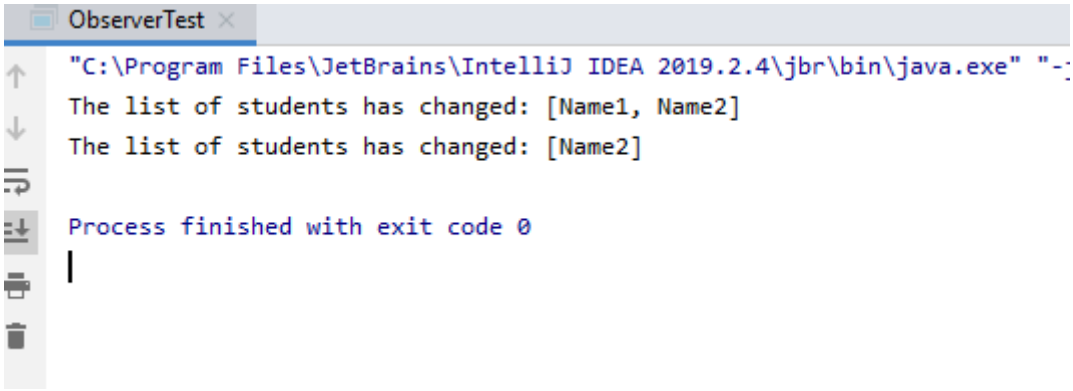

Рисунок 2 – Результат запуску тестового класу

2. Зберігач

Даний патерн проектування дозволяє зберігати поточний стан об’єкту, згодом можна буде відновити його стан, при цьому не порушує принцип інкапсуляції.

Програмна реалізація:

Створимо класс Користувач, класс, що зберігає дані користувачів, безпосередньо Зберігача, та тестовий клас.

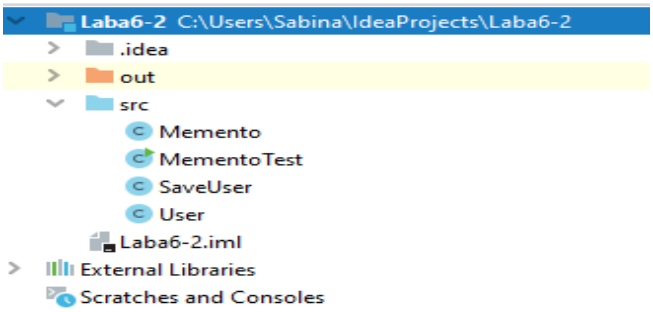

Рисунок 3 – Структура проекту (2)

Далі наведено лістинг коду проекту, що реалізує шаблон проектування Зберігач:

```
import java.util.ArrayList;
import java.util.List;
class Memento {
    private String name;
    private int age;
    public Memento(String name, int age){
        this.name = name;
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
}
class User {
    private String name;
    private int age;
    public User(String name, int age) {
        this.name = name;
        this.age = age;
        System.out.println(String.format("create: name = %s, age = %s", name,
age));
    }
    public Memento save(){
        System.out.println(String.format("save: name = %s, age = %s", name,
age));
        return new Memento(name, age);
    }
    public void restore(Memento memento){
        name = memento.getName();
        age = memento.getAge();
        System.out.println(String.format("restore: name = %s, age = %s",
name, age));
    }
}

class SaveUser {
    private List<Memento> list = new ArrayList<Memento>();
    public void add(Memento memento){
        list.add(memento);
    }
    public Memento get(int ind){
        return list.get(ind);
    }
}

public class MementoTest { // Test
    public static void main(String[] args) {
        SaveUser saveUser = new SaveUser();
        User user1 = new User("Peter", 17);
        User user2 = new User("Ian", 19);
```

```
        saveUser.add(user1.save());
        user1.restore(saveUser.get(0));
    }
}
```

Для перевірки роботи програми необхідно зкомпілювати та запустити тестовий клас.

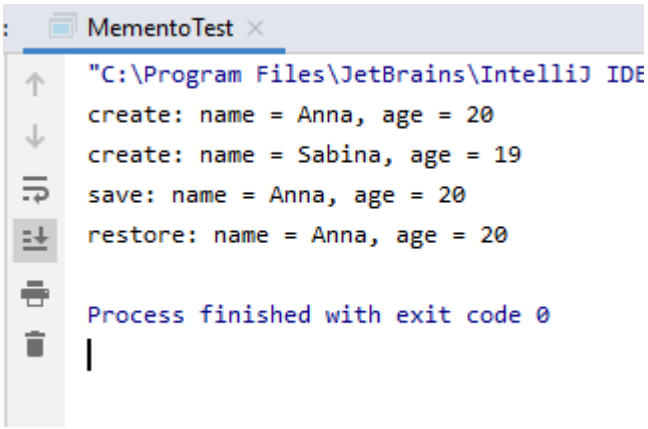

Рисунок 4 – Результат запуску тестового класу (2)

Висновки

У ході виконання даної лабораторної роботи було проведено ознайомлення з породжуючими, структурними та поведінковими патернами проектування, а також розроблена програмна реалізація двох поведінкових патернів: Спостерігач та Зберігач. Проекти були успішно запущені та відпрацювали без помилок, результати запусків приведені у лабораторній роботі.