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

Технологія розроблення програмного забезпечення

«Shell (total commander)»

Варіант 18

Виконав
студент групи ІА-13
Окаянченко Давид Олександрович

Перевірив:
Мягкий Михайло Юрійович

Київ 2023

Мета: Дослідити шаблони «ADAPTER», «BUILDER», «COMMAND», «CHAIN OF RESPONSIBILITY», «PROTOTYPE» та навчитися застосовувати один із них на практиці.

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:

18. Shell (total commander) (state, prototype, factory method, template method, interpreter, client-server)

Оболонка повинна вміти виконувати основні дії в системі - перегляд файлів папок в файлової системі, перемикання між дисками, копіювання,

видалення, переміщення об'єктів, пошук.

Хід роботи

Шаблони проєктування - це певні способи розв'язання типових проблем, які виникають під час розробки програмного забезпечення. Вони є своєрідними "рецептами" або наборами правил, які вже доведено було успішними в реальних проектах. Їх використання допомагає розробникам ефективно вирішувати спільні завдання та уникати типових помилок.

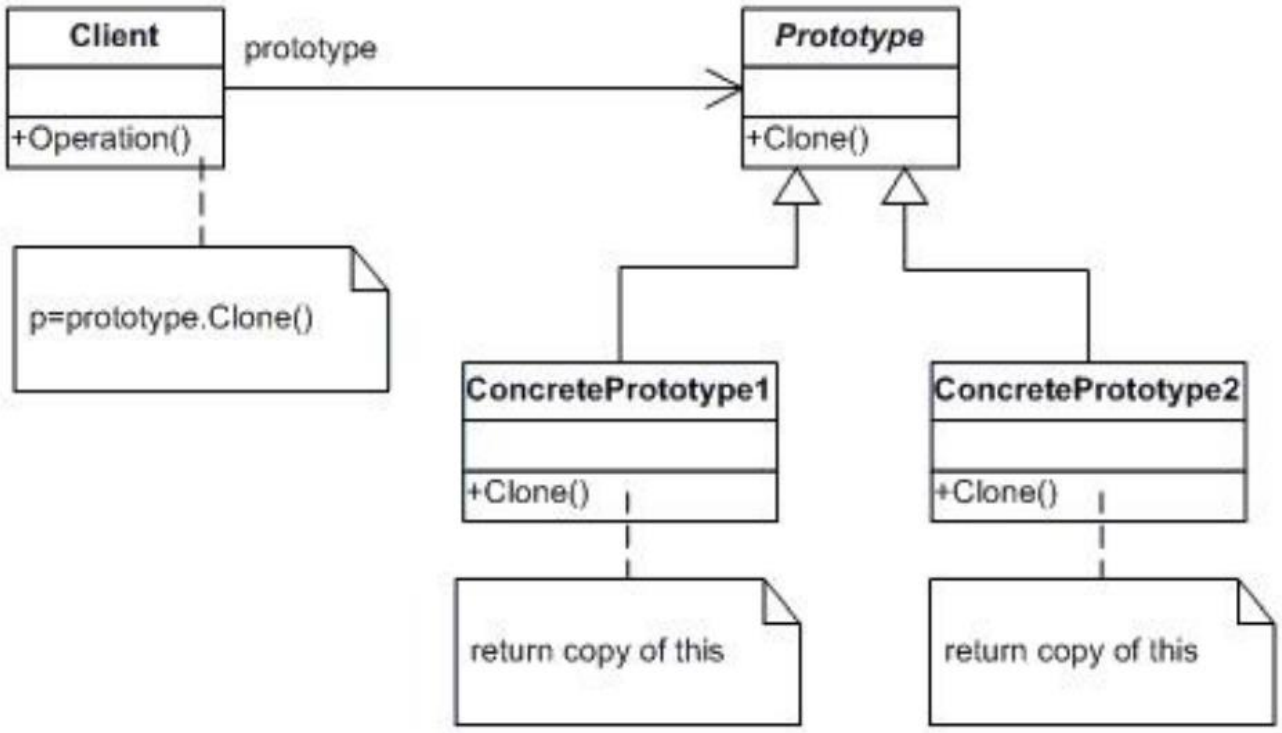
Важливі аспекти шаблонів проєктування:

- Полегшення розробки: Вони надають структурований підхід до розв'язання проблем, що допомагає розробникам швидше і ефективніше створювати програмне забезпечення.
- Підвищення якості: Шаблони допомагають уникати поширених помилок, що можуть призвести до поганої продуктивності або низької якості програми.
- Підвищення перевикористання: Вони сприяють створенню універсальних рішень, які можна використовувати в різних контекстах.
- Покращення зрозуміння: Використання шаблонів полегшує іншим розробникам розуміння коду та сприяє легшій підтримці.

- Спрощення спільної роботи: Шаблони допомагають командам розробників працювати спільно, оскільки вони знайомі із загальними концепціями та підходами.

Шаблон проєктування «Prototype»

Структура:



Призначення:

Шаблон "prototype" (прототип) використовується для створення об'єктів за "шаблоном" (чи "кресленню", "ескізу") шляхом копіювання шаблонного об'єкту. Для цього визначається метод "клонувати" в об'єктах цього класу. Цей шаблон зручно використати, коли заздалегідь відомо як виглядатиме кінцевий об'єкт (мінімізується кількість змін до об'єкту шляхом створення шаблону), а також для видалення необхідності створення об'єкту - створення відбувається за рахунок клонування, і зухвалій програмі абсолютно немає необхідності знати, як створювати об'єкт. Також, це дозволяє маніпулювати об'єктами під час виконання програми шляхом настроювання відповідних шаблонів; значно зменшується ієрархія спадкоємства (оскільки в іншому випадку це були б не шаблони, а вкладені класи, що наслідують).

Переваги та недоліки:

+ Дозволяє клонувати об’єкти без прив’язки до їхніх конкретних класів.

+ Менша кількість повторювань коду ініціалізації об’єктів.

+ Прискорює створення об’єктів.

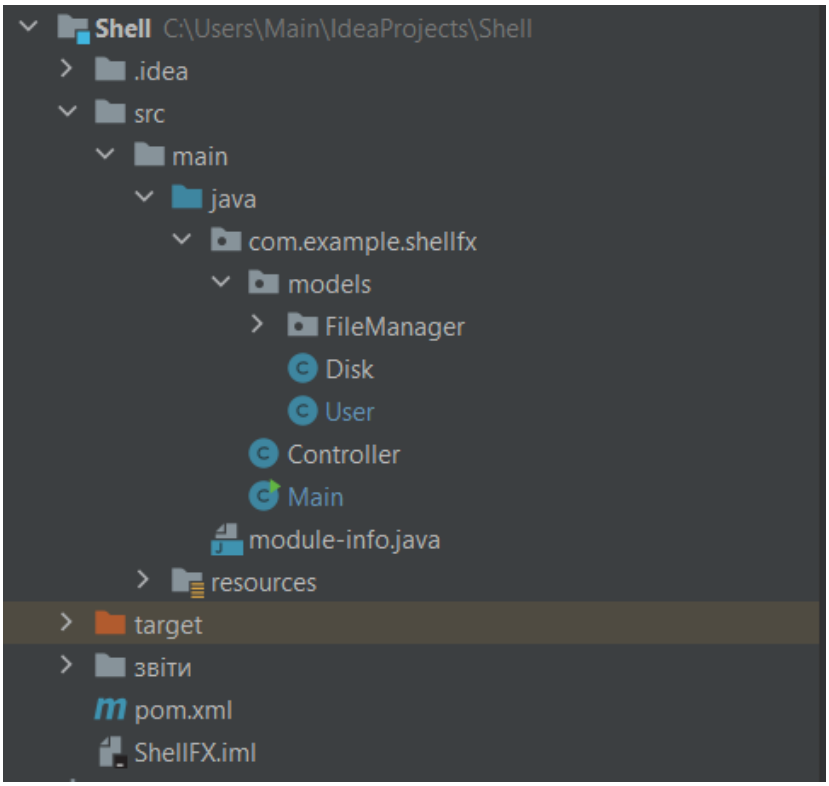
+ Альтернатива створенню підкласів під час конструювання складних об’єктів.

- Складно клонувати складові об’єкти, що мають посилання на інші об’єкти.

Реалізація:

Шаблон Прототип у моєму проєкті дозволяє створювати копії об'єктів користувачів на основі існуючих користувачів. Він забезпечує зручний спосіб створення нових користувачів з використанням вже наявних даних, що спрощує управління користувачами. Цей підхід допомагає уникнути дублювання коду та заощаджує час при створенні та редагуванні користувацьких облікових записів.

Стуктура проєкта:



Клас Main:

Цей клас є частиною додатку та використовується для створення графічного інтерфейсу користувацького файлового менеджера за допомогою JavaFX. Крім того, у методі "start" він створює два об'єкти класу "User" і демонструє реалізацію шаблону "Prototype", де "user2" створений як клон "user1", і виводить їхні дані на консоль, показуючи, що обидва об'єкти мають однакові характеристики, але вони не посилаються на один і той самий об'єкт.

```
package com.example.shellfx;

import com.example.shellfx.models.User;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;
import org.kordamp.bootstrapfx.BootstrapFX;
```

```
import java.io.IOException;

public class Main extends javafx.application.Application {
    public static void main(String[] args) {
        launch();
    }

    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(Main.class.getResource("shell.fxml"));
        Scene scene = new Scene(fxmlLoader.load());
        scene.getStylesheets().add(BootstrapFX.bootstrapFXStylesheet());
        stage.setTitle("Shell");
        stage.setScene(scene);
        stage.show();

        User user1 = new User("Davyd", "david@gmail.com", "123123");
        User user2 = user1.clone();

        System.out.println(user1);
        System.out.println(user2);
        System.out.println(user1 == user2);
    }
}
```

Клас User:

Цей клас представляє користувача і дозволяє зберігати інформацію про ім'я, електронну пошту та пароль користувача. Використання інтерфейсу "Cloneable" і методу "clone" в цьому класі надає можливість створювати копії користувачів для дублювання даних користувачів, якщо це необхідно.

У даному класі шаблон Prototype реалізований за допомогою інтерфейсу "Cloneable" та методу "clone". Цей метод дозволяє створювати глибокі копії об'єкта "User", включаючи всі його внутрішні дані. Якщо потрібно створити копію користувача, ми можемо використовувати метод "clone" для цього, що дозволяє створити новий об'єкт "User" з такими ж характеристиками, як і оригінал.

```
package com.example.shellfx.models;

import java.util.Objects;

public class User implements Cloneable {
    private String name;
    private String email;
    private String password;

    public User(String name, String email, String password) {
        this.name = name;
        this.email = email;
        this.password = password;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
```

```
    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    @Override
    public User clone() {
        try {
            return (User) super.clone();
        } catch (CloneNotSupportedException e) {
            throw new AssertionError();
        }
    }
}

    @Override
    public String toString() {
        return "User{" +
                "name='" + name + '\'' +
                ", email='" + email + '\'' +
                ", password='" + password + '\'' +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof User user)) return false;
        return Objects.equals(getName(), user.getName()) &&
Objects.equals(getEmail(), user.getEmail()) && Objects.equals(getPassword(),
user.getPassword());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getName(), getEmail(), getPassword());
    }
```

Висновок: У ході виконання лабораторної роботи було проведено ознайомлення з теоретичними відомостями та реалізовано шаблон проєктування «Prototype». Окрім того, підготовлений звіт включає всі необхідні компоненти, що відображають структуру розробленої системи.