****

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 №7  
ШАБЛОН «MEDIATOR», «FACADE»,

«BRIDGE», «TEMPLATE METHOD»

Виконав Перевірив:

студент групи ІА – 13: Мягкий М. Ю

Мельничук Дмитро

**Завдання:**

1. Ознайомитися з короткими теоретичними відомостями.

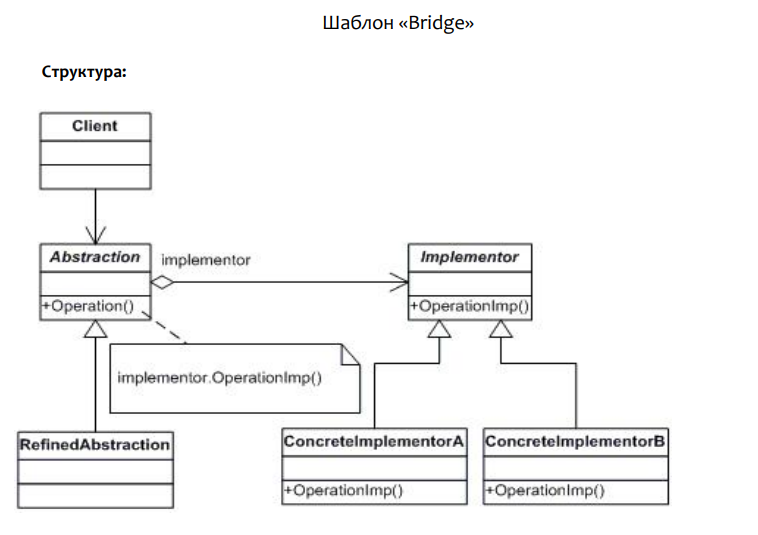
2. Реалізувати частину функціоналу робочої програми у вигляді класів та їх взаємодій для

досягнення конкретних функціональних можливостей.

3. Застосування одного з розглянутих шаблонів при реалізації програми.

**Хід роботи**

Шаблон «Bridge» (міст) використовується для поділу інтерфейсу і його реалізації. Це необхідно у випадках, коли може існувати кілька різних абстракцій, над якими можна проводити дії різними способами.



public interface Connection {

void establishConnection();

void sendData(String data);

void closeConnection();

}

// Реалізація абстракції - конкретний тип з'єднання (наприклад, Ethernet)

public class EthernetConnection implements Connection {

@Override

public void establishConnection() {

// Логіка встановлення з'єднання через Ethernet

}

@Override

public void sendData(String data) {

// Логіка передачі даних через Ethernet

}

@Override

public void closeConnection() {

// Логіка закриття з'єднання через Ethernet

}

}

// Інша реалізація абстракції - конкретний тип з'єднання (наприклад, Wi-Fi)

public class WifiConnection implements Connection {

@Override

public void establishConnection() {

// Логіка встановлення з'єднання через Wi-Fi

}

@Override

public void sendData(String data) {

// Логіка передачі даних через Wi-Fi

}

@Override

public void closeConnection() {

// Логіка закриття з'єднання через Wi-Fi

}

}

// Абстракція мережевого комунікатора

public abstract class Communicator {

protected Connection connection;

public Communicator(Connection connection) {

this.connection = connection;

}

public abstract void sendMessage(String message);

public abstract void initiateCall(String contact);

// Додаткові методи та функціонал для комунікатора

}

// Реалізація абстракції - конкретний мережевий комунікатор

public class OfficeCommunicator extends Communicator {

public OfficeCommunicator(Connection connection) {

super(connection);

}

@Override

public void sendMessage(String message) {

// Логіка відправки текстового повідомлення

connection.establishConnection();

connection.sendData(message);

connection.closeConnection();

}

@Override

public void initiateCall(String contact) {

// Логіка ініціювання голосового/відеодзвінка

connection.establishConnection();

// Логіка ініціювання дзвінка

connection.closeConnection();

}

// Додаткові методи та функціонал для комунікатора

}

public class Main {

public static void main(String[] args) {

// Використання Ethernet-з'єднання

Connection ethernetConnection = new EthernetConnection();

Communicator officeCommunicator1 = new OfficeCommunicator(ethernetConnection);

// Використання Wi-Fi-з'єднання

Connection wifiConnection = new WifiConnection();

Communicator officeCommunicator2 = new OfficeCommunicator(wifiConnection);

// Взаємодія з комунікаторами

officeCommunicator1.sendMessage("Привіт, як справи?");

officeCommunicator2.initiateCall("user1");

}