****

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 №5  
ШАБЛОНИ «ADAPTER», «BUILDER»,

«COMMAND», «CHAIN OF

RESPONSIBILITY», «PROTOTYPE»

Виконав Перевірив:

студент групи ІА – 13: Мягкий М. Ю

Мельничук Дмитро

**Завдання:**

1. Ознайомитися з короткими теоретичними відомостями.

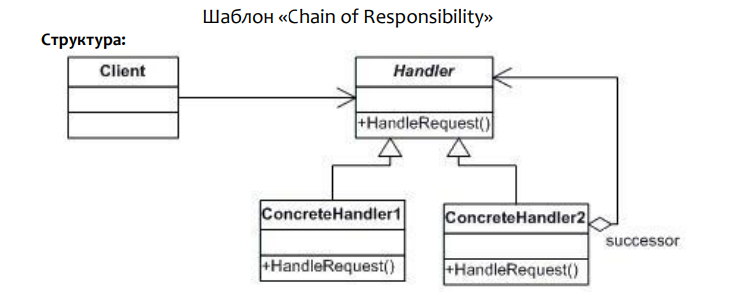
2. Реалізувати частину функціоналу робочої програми у вигляді класів та їх взаємодій для

досягнення конкретних функціональних можливостей.

3. Застосування одного з розглянутих шаблонів при реалізації програми.

**Хід роботи**

**Шаблон "Chain of Responsibility" дозволяє передавати запити послідовно через ланцюг обробників. Кожен обробник рішить, чи може він обробити запит, або передасть його наступному обробнику в ланцюгу.**



// Інтерфейс обробника

public interface CommunicationHandler {

void handleRequest(CommunicationRequest request);

}

// Конкретний обробник для відправлення текстових повідомлень

public class TextMessageHandler implements CommunicationHandler {

private CommunicationHandler nextHandler;

@Override

public void handleRequest(CommunicationRequest request) {

if (request.getType() == CommunicationRequestType.TEXT\_MESSAGE) {

// Логіка обробки текстового повідомлення

System.out.println("Text Message Handler processed the request.");

} else if (nextHandler != null) {

nextHandler.handleRequest(request);

}

}

public void setNextHandler(CommunicationHandler nextHandler) {

this.nextHandler = nextHandler;

}

}

// Конкретний обробник для відправлення файлів

public class FileHandler implements CommunicationHandler {

private CommunicationHandler nextHandler;

@Override

public void handleRequest(CommunicationRequest request) {

if (request.getType() == CommunicationRequestType.FILE) {

// Логіка обробки відправки файлу

System.out.println("File Handler processed the request.");

} else if (nextHandler != null) {

nextHandler.handleRequest(request);

}

}

public void setNextHandler(CommunicationHandler nextHandler) {

this.nextHandler = nextHandler;

}

}

// Клас запиту для використання у ланцюгу обробників

public class CommunicationRequest {

private CommunicationRequestType type;

public CommunicationRequest(CommunicationRequestType type) {

this.type = type;

}

public CommunicationRequestType getType() {

return type;

}

}

// Перерахування для типів запитів

public enum CommunicationRequestType {

TEXT\_MESSAGE,

FILE

}

public class Main {

public static void main(String[] args) {

// Створення ланцюгу обробників

CommunicationHandler textMessageHandler = new TextMessageHandler();

CommunicationHandler fileHandler = new FileHandler();

textMessageHandler.setNextHandler(fileHandler);

// Створення та відправка запиту

CommunicationRequest textMessageRequest = new CommunicationRequest(CommunicationRequestType.TEXT\_MESSAGE);

textMessageHandler.handleRequest(textMessageRequest);

CommunicationRequest fileRequest = new CommunicationRequest(CommunicationRequestType.FILE);

textMessageHandler.handleRequest(fileRequest);

}

}