****

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 №8  
ШАБЛОНИ «COMPOSITE»,

«FLYWEIGHT», «INTERPRETER»,

«VISITOR»

Виконав Перевірив:

студент групи ІА – 13: Мягкий М. Ю

Мельничук Дмитро

**Завдання:**

1. Ознайомитися з короткими теоретичними відомостями.

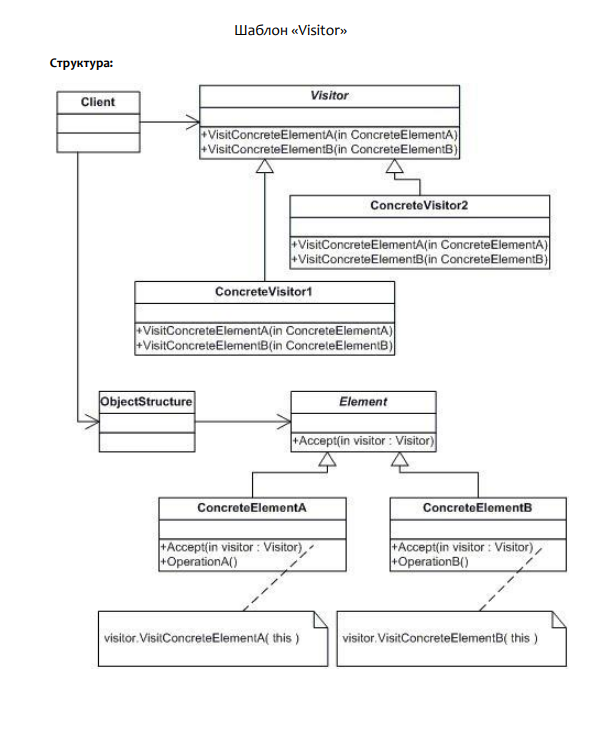
2. Реалізувати частину функціоналу робочої програми у вигляді класів і їх взаємодій для

досягнення конкретних функціональних можливостей.

3. Застосування одного з розглянутих шаблонів при реалізації програми.

**Хід роботи**

**Шаблон відвідувач дозволяє вказувати операції над елементами без зміни структури конкретних елементів. Таким чином вкрай зручно додавати нові операції, проте дуже важко додавати нові елементи в ієрархію (необхідно додавати відповідні методи для обробки їх відвідувань в кожного відвідувача).**



public interface ContactVisitor {

void visit(UserContact userContact);

void visit(GroupContact groupContact);

}

// Клас UserContact, який може бути відвіданий

public class UserContact {

private String name;

public UserContact(String name) {

this.name = name;

}

public String getName() {

return name;

}

public void accept(ContactVisitor visitor) {

visitor.visit(this);

}

}

// Клас GroupContact, який може бути відвіданий

public class GroupContact {

private String groupName;

public GroupContact(String groupName) {

this.groupName = groupName;

}

public String getGroupName() {

return groupName;

}

public void accept(ContactVisitor visitor) {

visitor.visit(this);

}

}

// Клас Communicator, який використовує Visitor для операцій над контактами

public class Communicator {

private List<Object> contacts = new ArrayList<>();

public void addContact(Object contact) {

contacts.add(contact);

}

public void performOperations(ContactVisitor visitor) {

for (Object contact : contacts) {

if (contact instanceof UserContact) {

((UserContact) contact).accept(visitor);

} else if (contact instanceof GroupContact) {

((GroupContact) contact).accept(visitor);

}

}

}

}

// Реалізація конкретного Visitor

public class DisplayVisitor implements ContactVisitor {

@Override

public void visit(UserContact userContact) {

System.out.println("Displaying user contact: " + userContact.getName());

}

@Override

public void visit(GroupContact groupContact) {

System.out.println("Displaying group contact: " + groupContact.getGroupName());

}

}

public class Main {

public static void main(String[] args) {

Communicator communicator = new Communicator();

communicator.addContact(new UserContact("User1"));

communicator.addContact(new GroupContact("Team"));

ContactVisitor displayVisitor = new DisplayVisitor();

communicator.performOperations(displayVisitor);

}

}