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 № 6

Технології розроблення програмного забезпечення

«Патерни проектування.»

Тема: IRC-Клієнт

Виконала:
студентка групи ІА-32
Красоха В.О.

Перевірив:
Мягкий М.Ю.

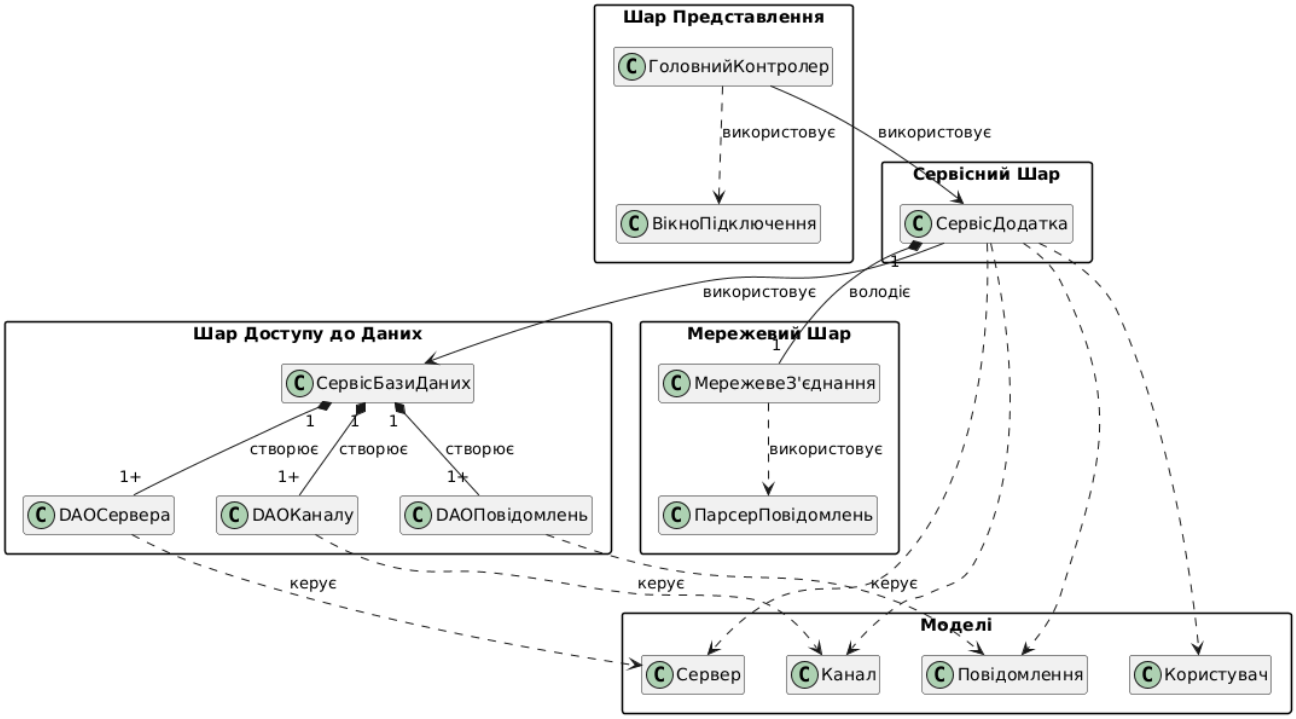
Київ 2025

Тема: Патерни проектування.

Мета: Вивчити структуру шаблонів «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator» та навчитися застосовувати їх в реалізації програмної системи.

Хід роботи

Діаграма класів:



Патерн Observer (Спостерігач)

Це найважливіший патерн для будь-якого чат-додатка. Він використовується у взаємодії між Мережевим шаром та Інтерфейсом (UI). Коли Мережеве З'єднання отримує нове повідомлення від IRC-сервера, воно не знає, як його малювати на екрані. Воно просто "сповіщає" підписані на нього об'єкти (Контролери). IRCConnection виступає як Subject (Суб'єкт), а MainController — як Observer (Спостерігач). Метод displayMessage викликається щоразу, коли приходить нова подія з мережі.

Патерн Factory Method (Фабричний метод)

Цей патерн використовується для створення об'єктів доступу до даних (DAO). Його можна помітити у класі DatabaseService (або ServiceБазиДаних на

діаграмі). Замість того, щоб створювати MessageDao чи ServerDao вручну в різних частинах програми, я використовую DatabaseService, який "штампує" (створює) ці об'єкти, передаючи їм готове з'єднання з БД. На діаграмі класів чітко видно стрілки створює від СервісБазиДаних до DAOServera, DAOKanalu та DAOPovidomlen. Це і є логіка фабрики — один об'єкт відповідає за створення інших.

Доданий код:

ConnectDialogController.java:

```
package org.ircclient.controller;

import javafx.fxml.FXML;
import javafx.scene.control.TextField;
import org.ircclient.model.Server;

public class ConnectDialogController {
    @FXML
    private TextField serverNameField;
    @FXML
    private TextField hostField;
    @FXML
    private TextField portField;
    @FXML
    private TextField nicknameField;

    @FXML
    public void initialize() {
        portField.setText("6667");
        nicknameField.setText("IRCUser");
    }

    public Server getServer() {
        String name = serverNameField.getText().trim();
        String host = hostField.getText().trim();
        String portStr = portField.getText().trim();
        String nickname = nicknameField.getText().trim();

        if (host.isEmpty() || portStr.isEmpty() || nickname.isEmpty()) {
            return null;
        }

        try {
            int port = Integer.parseInt(portStr);
            if (name.isEmpty()) {
                name = host + ":" + port;
            }
            return new Server(name, host, port, nickname);
        } catch (NumberFormatException e) {
            return null;
        }
    }
}
```

MainController.java:

```
package org.ircclient.controller;

import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.control.*;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.scene.Node;
import javafx.scene.Parent;
import org.ircclient.model.Channel;
import org.ircclient.model.Message;
import org.ircclient.model.Server;
import org.ircclient.model.User;
import org.ircclient.service.DatabaseService;
import org.ircclient.service.IRCService;

import java.io.IOException;
import java.time.format.DateTimeFormatter;
import java.util.List;
import java.util.Map;

public class MainController {
    @FXML
    private ListView<String> channelsList;
    @FXML
    private TabPane chatTabs;
    @FXML
    private TextField messageInput;
    @FXML
    private ListView<String> usersList;
    @FXML
    private Label usersLabel;

    private IRCService ircService;
    private DatabaseService databaseService;
    private String currentChannel;
    private Map<String, Tab> channelTabs;
    private Map<String, TextArea> channelChatAreas;

    @FXML
    public void initialize() {
        databaseService = new DatabaseService();
        ircService = new IRCService(databaseService);
        channelTabs = new java.util.concurrent.ConcurrentHashMap<>();
        channelChatAreas = new java.util.concurrent.ConcurrentHashMap<>();

        setupIRCCallbacks();
        loadChannelsFromDB();

        messageInput.setOnKeyPressed(this::handleKeyPress);
    }


    private void setupIRCCallbacks() {
        ircService.setOnMessageReceived(this::onMessageReceived);
        ircService.setOnChannelJoined(this::onChannelJoined);
        ircService.setOnChannelLeft(this::onChannelLeft);
```

```
    ircService.setOnUserListUpdated(this::onUserListUpdated);
    ircService.setOnChannelListReceived(this::onChannelListReceived);
    ircService.setOnError(this::onError);
    ircService.setOnSystemMessage(this::onSystemMessage);
}

/**
 * Обробка натискання клавіші
 */
private void handleKeyPress(KeyEvent event) {
    if (event.getCode() == KeyCode.ENTER) {
        handleSendMessage();
    }
}

/**
 * Підключення до сервера
 */
@FXML
private void handleConnect() {
    try {
        Dialog<Server> dialog = new Dialog<>();
        dialog.setTitle("Підключення до сервера");
        dialog.setHeaderText("Введіть дані для підключення");

        FXMLLoader loader = new FXMLLoader(getClass().getResource("/fxml/connect-dialog.fxml"));
        dialog.getDialogPane().setContent(loader.load());
        ConnectDialogController controller = loader.getController();

        dialog.getDialogPane().getButtonTypes().addAll(ButtonType.OK, ButtonType.CANCEL);

        dialog.setResultConverter(buttonType -> {
            if (buttonType == ButtonType.OK) {
                return controller.getServer();
            }
            return null;
        });

        dialog.showAndWait().ifPresent(server -> {
            if (server != null) {
                try {
                    ircService.connect(server);
                    addSystemMessage("Підключення до " + server.getHost() + ":" + server.getPort() + "...");
                } catch (IOException e) {
                    showError("Помилка підключення: " + e.getMessage());
                }
            }
        });
    } catch (IOException e) {
        showError("Помилка завантаження діалогу: " + e.getMessage());
    }
}

/**
 * Відключення від сервера
 */
@FXML
private void handleDisconnect() {
    ircService.disconnect();
    channelTabs.clear();
    channelChatAreas.clear();
    chatTabs.getTabs().clear();
    channelsList.getItems().clear();
```

```
    usersList.getItems().clear();
    addSystemMessage("Відключено від сервера");
}

/**
 * Відправка повідомлення
 */
@FXML
private void handleSendMessage() {
    String text = messageInput.getText().trim();
    if (text.isEmpty()) {
        return;
    }

    Tab selectedTab = chatTabs.getSelectionModel().getSelectedItem();
    if (selectedTab == null) {
        showError("Виберіть канал для відправки повідомлення");
        return;
    }

    String target = selectedTab.getText();

    try {
        if (text.startsWith("/")) {
            // Команда IRC
            ircService.executeCommand(text);
        } else {
            // Звичайне повідомлення
            ircService.sendMessage(target, text);
            // Відображення свого повідомлення
            displayMessage(target, new Message(ircService.getCurrentServer().getNickname(), text));
        }
        messageInput.clear();
    } catch (IOException e) {
        showError("Помилка відправки: " + e.getMessage());
    }
}

/**
 * Приєднання до каналу
 */
@FXML
private void handleJoinChannel() {
    TextInputDialog dialog = new TextInputDialog();
    dialog.setTitle("Приєднання до каналу");
    dialog.setHeaderText("Введіть назву каналу");
    dialog.setContentText("Канал:");

    dialog.showAndWait().ifPresent(channelName -> {
        if (!channelName.isEmpty()) {
            try {
                ircService.joinChannel(channelName);
            } catch (IOException e) {
                showError("Помилка приєднання: " + e.getMessage());
            }
        }
    });
}

/**
 * Створення нового каналу
 */
@FXML
```

```
private void handleCreateChannel() {
    handleJoinChannel(); // Те саме що приєднання
}

/**
 * Оновлення списку каналів
 */
@FXML
private void handleRefreshChannelList() {
    if (!ircService.isConnected()) {
        showError("Не підключено до сервера");
        return;
    }
    try {
        ircService.executeCommand("/list");
    } catch (IOException e) {
        showError("Помилка оновлення списку: " + e.getMessage());
    }
}

/**
 * Вибір каналу зі списку
 */
@FXML
private void handleChannelSelect() {
    String selected = channelsList.getSelectionModel().getSelectedItem();
    if (selected != null) {
        Tab tab = channelTabs.get(selected);
        if (tab != null) {
            chatTabs.getSelectionModel().select(tab);
        }
    }
}

/**
 * Вихід з програми
 */
@FXML
private void handleExit() {
    ircService.disconnect();
    databaseService.close();
    Platform.exit();
}

/**
 * Про програму
 */
@FXML
private void handleAbout() {
    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.setTitle("Про програму");
    alert.setHeaderText("IRC Клієнт");
    alert.setContentText("IRC клієнт на Java з використанням JavaFX\nВерсія 1.0");
    alert.showAndWait();
}

/**
 * Callback: отримано повідомлення
 */
private void onMessageReceived(Message message) {
    Platform.runLater(() -> {
        String channelName = message.getChannelName();
        if (channelName == null) {
```

```
            // Fallback: використовуємо поточний канал або sender
            channelName = message.getIsPrivate() ? message.getSender() : currentChannel;
        }

        if (channelName != null) {
            displayMessage(channelName, message);
        }
    });
}

/**
 * Callback: приєднано до каналу
 */
private void onChannelJoined(Channel channel) {
    Platform.runLater(() -> {
        String channelName = channel.getName();
        createChannelTab(channelName);
        channelsList.getItems().add(channelName);
        addSystemMessage("Приєднано до каналу: " + channelName);
        loadChannelHistory(channel);
    });
}

/**
 * Callback: вийшли з каналу
 */
private void onChannelLeft(String channelName) {
    Platform.runLater(() -> {
        Tab tab = channelTabs.remove(channelName);
        if (tab != null) {
            chatTabs.getTabs().remove(tab);
        }
        channelChatAreas.remove(channelName);
        channelsList.getItems().remove(channelName);
        addSystemMessage("Вийшли з каналу: " + channelName);
    });
}

/**
 * Callback: оновлено список користувачів
 */
private void onUserListUpdated(String channelName) {
    Platform.runLater(() -> {
        if (chatTabs.getSelectionModel().getSelectedItem() != null &&
            chatTabs.getSelectionModel().getSelectedItem().getText().equals(channelName)) {
            updateUsersList(channelName);
        }
    });
}

/**
 * Callback: отримано список каналів
 */
private void onChannelListReceived(List<String> channels) {
    Platform.runLater(() -> {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Список каналів");
        alert.setHeaderText("Доступні канали (" + channels.size() + "):");

        // 1. Створюємо StringBuilder, як і раніше
        StringBuilder content = new StringBuilder();
        for (String channel : channels) {
            content.append(channel).append("\n");
```

```
        }

        // --- ПОКРАЩЕННЯ ПОЧИНАЄТЬСЯ ТУТ ---

        // 2. Створюємо TextArea для відображення списку
        TextArea textArea = new TextArea(content.toString());
        textArea.setEditable(false); // Робимо його лише для читання
        textArea.setWrapText(true);  // Увімкнемо перенос тексту

        // 3. (Важливо) Обмежуємо розмір TextArea, щоб Alert не став гігантським
        textArea.setPrefSize(600, 400); // 600px ширина, 400px висота

        // 4. Додаємо TextArea у DialogPane
        // НЕ ВИКОРИСТОВУЙТЕ alert.setContentText(...)
        alert.getDialogPane().setContent(textArea);

        // 5. (Бонус) Дозволяємо користувачу змінювати розмір вікна
        alert.setResizable(true);

        // --- КІНЕЦЬ ПОКРАЩЕННЯ ---

        alert.showAndWait();
    });
}

/**
 * Callback: помилка
 */
private void onError(String error) {
    Platform.runLater(() -> showError(error));
}

/**
 * Callback: системне повідомлення
 */
private void onSystemMessage(String message) {
    Platform.runLater(() -> addSystemMessage(message));
}

/**
 * Створення вкладки для каналу
 */
private void createChannelTab(String channelName) {
    if (channelTabs.containsKey(channelName)) {
        return;
    }

    try {
        Tab tab = new Tab(channelName);
        FXMLLoader loader = new FXMLLoader(getClass().getResource("/fxml/channel-tab.fxml"));
        SplitPane content = loader.load();

        // Знаходимо TextArea через пошук у всьому дереві
        TextArea chatArea = findTextArea(content);
        if (chatArea == null) {
            // Якщо не знайдено, створюємо вручну
            ScrollPane scrollPane = new ScrollPane();
            scrollPane.setFitToWidth(true);
            scrollPane.setFitToHeight(true);
            chatArea = new TextArea();
            chatArea.setEditable(false);
            chatArea.setWrapText(true);
            chatArea.setId("chatArea");
```

```
            scrollPane.setContent(chatArea);
            content.getItems().set(0, scrollPane);
        }

        channelChatAreas.put(channelName, chatArea);
        tab.setContent(content);
        tab.setOnSelectionChanged(e -> {
            if (tab.isSelected()) {
                currentChannel = channelName;
                updateUsersList(channelName);
            }
        });

        channelTabs.put(channelName, tab);
        chatTabs.getTabs().add(tab);
        chatTabs.getSelectionModel().select(tab);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

/**
 * Пошук TextArea у дереві компонентів
 */
private TextArea findTextArea(Node node) {
    if (node instanceof TextArea) {
        return (TextArea) node;
    }
    if (node instanceof Parent) {
        for (Node child : ((Parent) node).getChildrenUnmodifiable()) {
            TextArea result = findTextArea(child);
            if (result != null) {
                return result;
            }
        }
    }
    return null;
}

/**
 * Відображення повідомлення в чаті
 */
private void displayMessage(String channelName, Message message) {
    TextArea chatArea = channelChatAreas.get(channelName);
    if (chatArea == null) {
        createChannelTab(channelName);
        chatArea = channelChatAreas.get(channelName);
    }

    if (chatArea != null) {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("HH:mm:ss");
        String timestamp = message.getTimestamp().format(formatter);
        String formattedMessage = formatMessage(message, timestamp);
        chatArea.appendText(formattedMessage + "\n");
        // Автоскрол
        chatArea.setScrollTop(Double.MAX_VALUE);
    }
}

/**
 * Форматування повідомлення з кольорами
 */
private String formatMessage(Message message, String timestamp) {
```

```
        String sender = message.getSender() != null ? message.getSender() : "System";
        String content = message.getContent();

        switch (message.getType()) {
            case SYSTEM:
                return String.format("[%s] *** %s", timestamp, content);
            case ERROR:
                return String.format("[%s] !!! %s", timestamp, content);
            case JOIN:
                return String.format("[%s] >>> %s %s", timestamp, sender, content);
            case PART:
            case QUIT:
                return String.format("[%s] <<< %s %s", timestamp, sender, content);
            case TOPIC:
                return String.format("[%s] *** Тема: %s", timestamp, content);
            default:
                return String.format("[%s] <%s> %s", timestamp, sender, content);
        }
    }

    /**
     * Оновлення списку користувачів
     */
    private void updateUsersList(String channelName) {
        List<User> users = ircService.getChannelUsers(channelName);
        ObservableList<String> userNames = FXCollections.observableArrayList();
        for (User user : users) {
            userNames.add(user.toString());
        }
        usersList.setItems(userNames);
        usersLabel.setText("Користувачі (" + users.size() + ")");
    }

    /**
     * Завантаження каналів з БД
     */
    private void loadChannelsFromDB() {
        // Можна завантажити збережені канали
    }

    /**
     * Завантаження історії каналу
     */
    private void loadChannelHistory(Channel channel) {
        try {
            List<org.ircclient.model.Message> messages =
databaseService.getMessageDao().findByChannelId(channel.getId());
            for (org.ircclient.model.Message msg : messages) {
                displayMessage(channel.getName(), msg);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * Додавання системного повідомлення
     */
    private void addSystemMessage(String message) {
        Message sysMsg = new Message(null, message);
        sysMsg.setType(Message.MessageType.SYSTEM);
        if (currentChannel != null) {
            displayMessage(currentChannel, sysMsg);
```

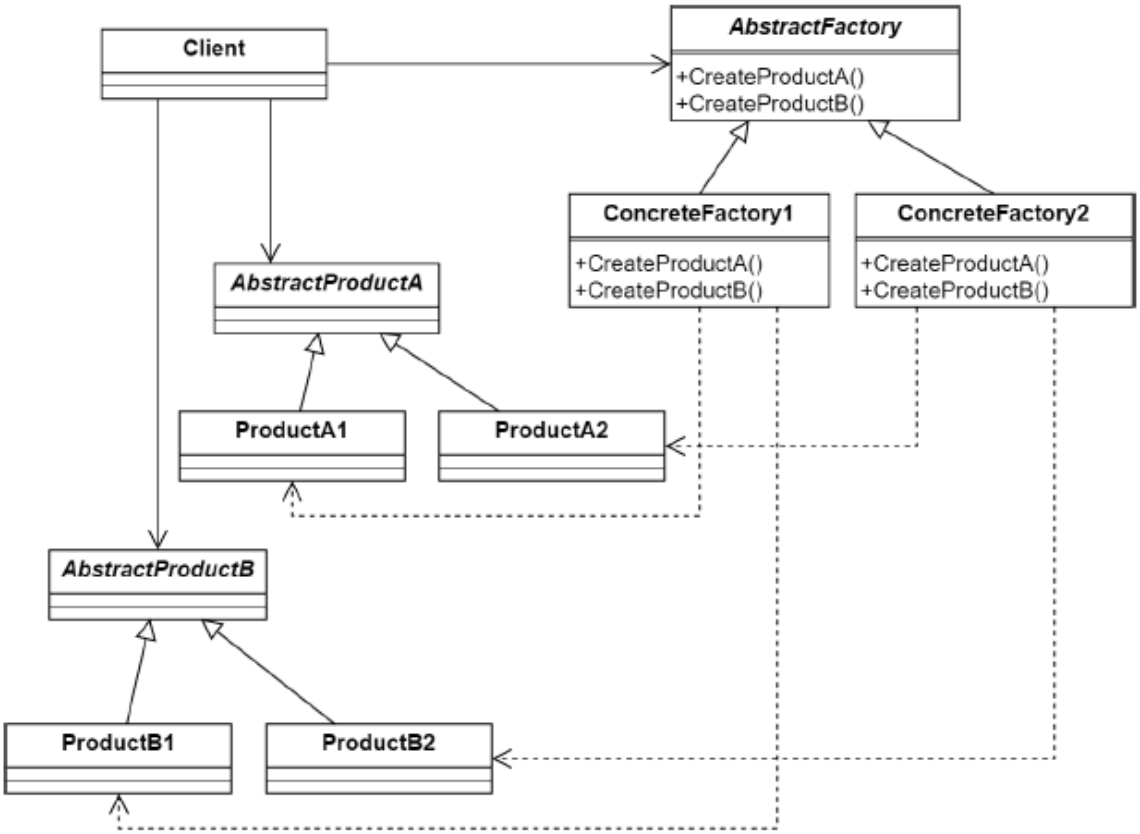
```
        }
    }

    /**
     * Показ помилки
     */
    private void showError(String message) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Помилка");
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}
```

Контрольні питання:

1. Яке призначення шаблону «Абстрактна фабрика»?

Шаблон Abstract Factory (Абстрактна фабрика) призначений для створення сімейств взаємопов'язаних або залежних об'єктів без прив'язки до конкретних класів.

2. Нарисуйте структуру шаблону «Абстрактна фабрика».

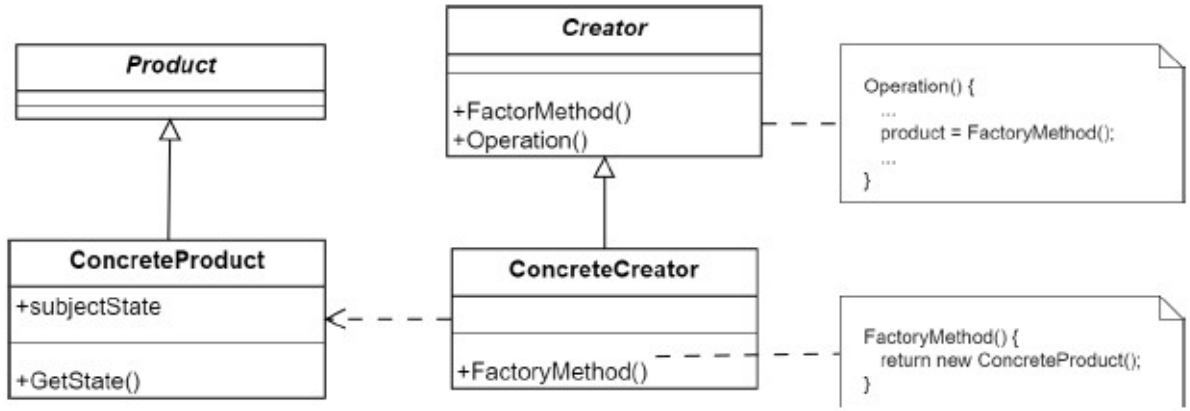


3. Які класи входять в шаблон «Абстрактна фабрика», та яка між ними взаємодія?

- AbstractFactory — інтерфейс для створення абстрактних продуктів
- ConcreteFactory — реалізація, яка створює конкретні продукти
- AbstractProduct — інтерфейс для продуктів
- ConcreteProduct — конкретна реалізація продуктів Клієнт взаємодіє з AbstractFactory і AbstractProduct, не знаючи конкретних класів; ConcreteFactory створює ConcreteProduct.

4. Яке призначення шаблону «Фабричний метод»? Шаблон Factory Method (Фабричний метод) призначений для створення об’єкта через метод, не вказуючи його конкретний клас, делегуючи це підкласам.

5. Нарисуйте структуру шаблону «Фабричний метод».



6. Які класи входять в шаблон «Фабричний метод», та яка між ними взаємодія?

- Creator — абстрактний клас або інтерфейс з фабричним методом
- ConcreteCreator — підклас, що реалізує фабричний метод
- Product — інтерфейс або абстрактний клас продукту

- ConcreteProduct — конкретний продукт Клієнт працює з Creator і Product; ConcreteCreator створює ConcreteProduct через фабричний метод.

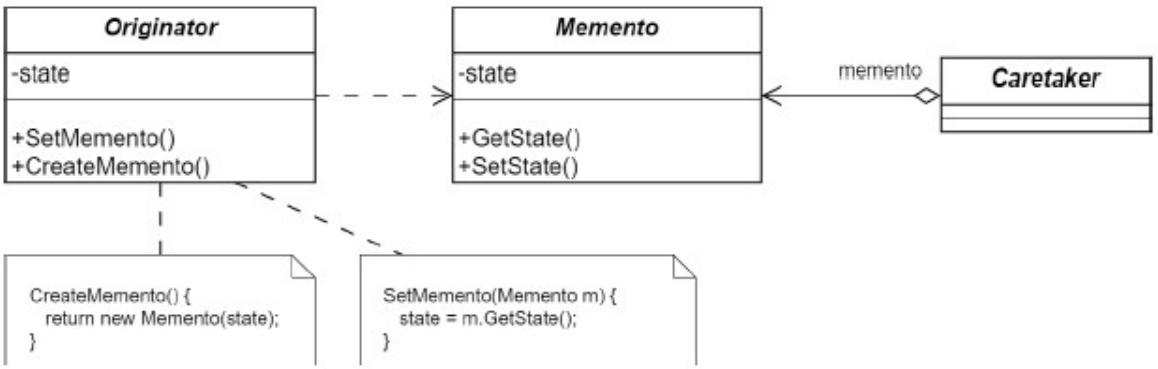
7. Чим відрізняється шаблон «Абстрактна фабрика» від «Фабричний метод»?

- Фабричний метод створює один об’єкт через метод, делегуючи підкласу
- Абстрактна фабрика створює сімейство взаємопов’язаних об’єктів, не прив’язуючись до конкретних класів
- Абстрактна фабрика використовує кілька фабричних методів для різних продуктів, а фабричний метод — один

8. Яке призначення шаблону «Знімок»?

Шаблон Memento (Знімок) дозволяє зберегти внутрішній стан об’єкта та відновити його пізніше, не порушуючи інкапсуляцію.

9. Нарисуйте структуру шаблону «Знімок».



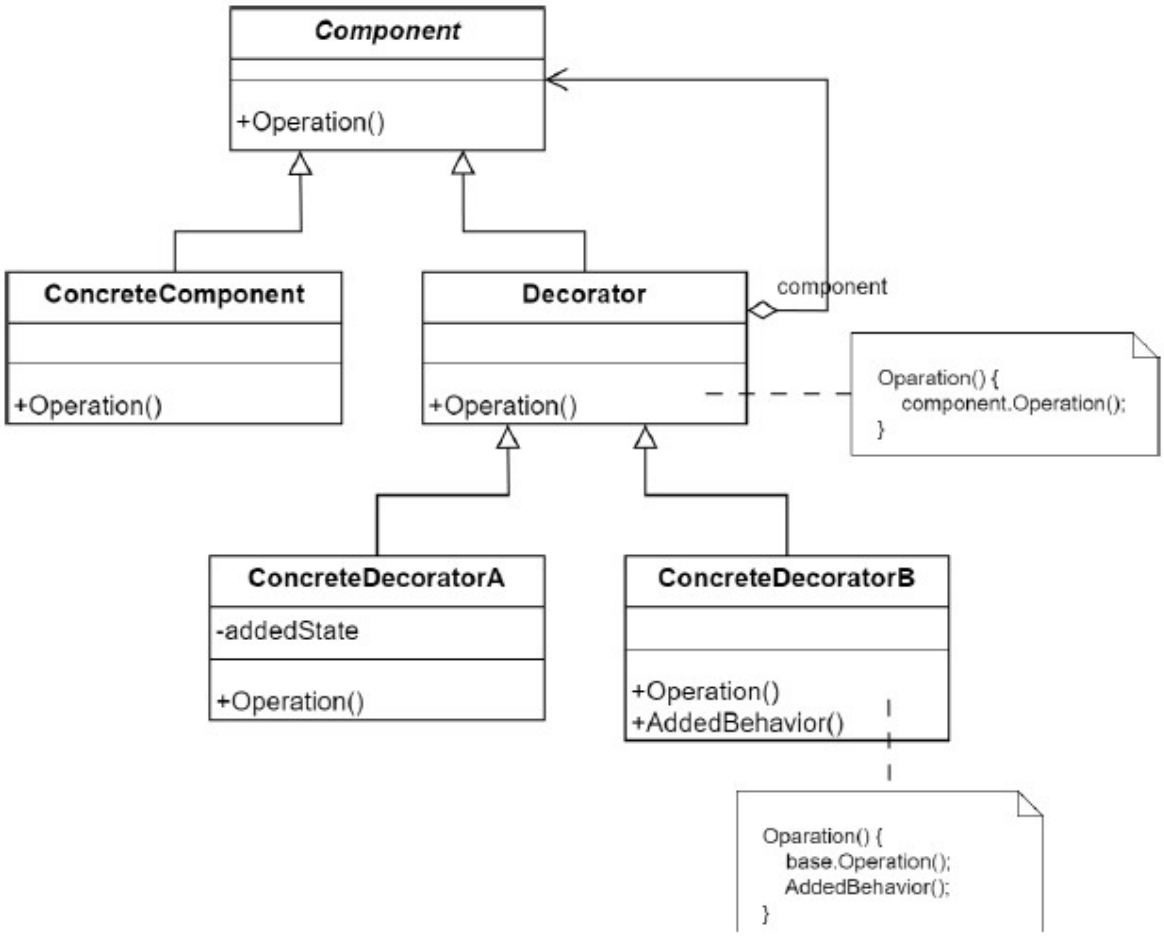
10. Які класи входять в шаблон «Знімок», та яка між ними взаємодія?

- Originator — об’єкт, стан якого потрібно зберегти
- Memento — зберігає внутрішній стан Originator
- Caretaker — керує збереженням і відновленням Memento Originator створює Memento і передає його Caretaker; Caretaker зберігає Memento та передає його назад для відновлення стану.

11. Яке призначення шаблону «Декоратор»?

Шаблон Decorator (Декоратор) дозволяє динамічно додавати об’єкту нову поведінку або функціональність, обгортаючи його у допоміжні об’єкти.

12. Нарисуйте структуру шаблону «Декоратор».



13. Які класи входять в шаблон «Декоратор», та яка між ними взаємодія?

- Component — базовий інтерфейс або абстрактний клас
- ConcreteComponent — конкретна реалізація об’єкта
- Decorator — абстрактний клас, що містить посилання на Component
- ConcreteDecorator — реалізація декоратора, додає поведінку Decorator делегує виклики ConcreteComponent, при цьому ConcreteDecorator розширює або змінює поведінку.

14. Які є обмеження використання шаблону «Декоратор»?

- Може ускладнити структуру через велику кількість об’єктів-декораторів
- Декоратори та Component повинні мати однаковий інтерфейс
- Може бути важко відслідковувати порядок декорацій і взаємодії
- Не підходить для об’єктів із сильно фіксованою поведінкою

Висновок: Під час виконання лабораторної роботи я вивчила структуру шаблонів «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator» та навчилася застосовувати їх в реалізації програмної системи.