Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота № 6

З дисципліни «Технології розроблення програмного забезпечення»

Тема: «Шаблони «Abstract Factory»,
«Factory Method», «Memento»,
«Observer», «Decorator»»

Виконала: Лапа Руслана Ігорівна

студент групи ІА-11

Дата здачі

Захищено з балом

Перевірив:

ст. вик. кафедри ІСТ

Колеснік В. М.

Київ 2023

Тема: Шаблони «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator».

Мета: реалізувати один з розглянутих шаблонів.

Хід роботи:

..8 Powershell terminal (strategy, command, factory method, bridge, interpreter, client-server)

Термінал для powershell повинен нагадувати типовий термінал з можливістю налаштування кольорів синтаксичних конструкцій, розміру вікна, фону вікна, а також виконання команд powershell і виконуваних файлів, а також працювати в декількох вікнах терміналу (у вкладках або одночасно шляхом розділення вікна).

1. Реалізувати не менше 3-х класів відповідно до обраної теми

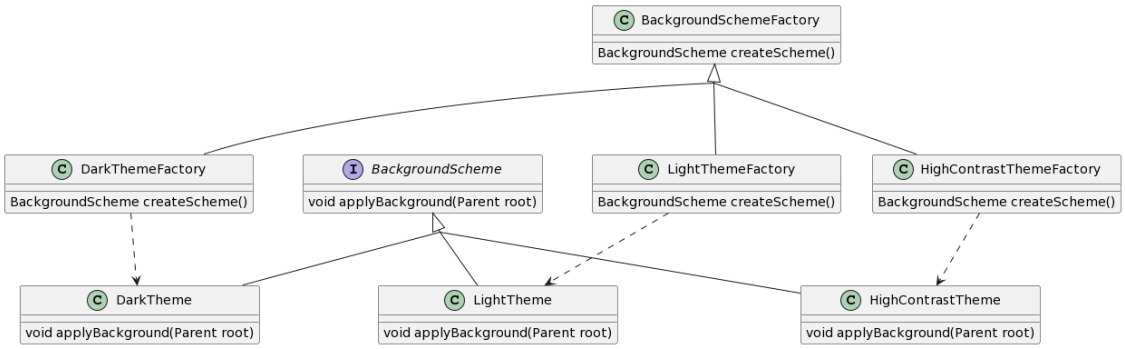

Рис. 1 – Діаграма послідовностей для нижче описаного коду

2. Реалізація шаблону «Factory Method»

```
package com.example.terminal.Model.Themes;

import javafx.scene.Parent;
3 implementations
public interface BackgroundTheme {
    1 usage   3 implementations
    void applyBackground(Parent root);
}
```

```java
package com.example.terminal.Model.Themes;

3 implementations
public interface BackgroundThemeFactory {
    1 usage   3 implementations
    BackgroundTheme createTheme();
}
```

```java
package com.example.terminal.Model.Themes;


import javafx.scene.Parent;

public class DarkTheme implements BackgroundTheme {
    1 usage
    @Override
    public void applyBackground(Parent root) { root.setStyle("-fx-background-color: #222222;"); }
}
```

```java
package com.example.terminal.Model.Themes;

public class DarkThemeFactory implements BackgroundThemeFactory {
    1 usage
    @Override
    public BackgroundTheme createTheme() { return new DarkTheme(); }
}
```

```java
package com.example.terminal.Model.Themes;


import javafx.scene.Parent;

public class HighContrastTheme implements BackgroundTheme {
    1 usage
    @Override
    public void applyBackground(Parent root) { root.setStyle("-fx-background-color: #000000;"); }
}
```

```java
package com.example.terminal.Model.Themes;

public class HighContrastThemeFactory implements BackgroundThemeFactory {
    1 usage
    @Override
    public BackgroundTheme createTheme() { return new HighContrastTheme(); }
}
```

```java
package com.example.terminal.Model.Themes;


import javafx.scene.Parent;

public class LightTheme implements BackgroundTheme {
    1 usage
    @Override
    public void applyBackground(Parent root) { root.setStyle("-fx-background-color: #FFFFFF;"); }
}
```

```java
package com.example.terminal.Model.Themes;

public class LightThemeFactory implements BackgroundThemeFactory {
    1 usage
    @Override
    public BackgroundTheme createTheme() { return new LightTheme(); }
}
```

```java
package com.example.terminal.Model.Themes;

import java.util.HashMap;
import java.util.Map;

public class ThemeManager {
    4 usages
    private static final Map<String, BackgroundThemeFactory> factories = new HashMap<>();

    static {
        factories.put("dark", new DarkThemeFactory());
        factories.put("light", new LightThemeFactory());
        factories.put("highcontrast", new HighContrastThemeFactory());
    }

    1 usage
    public static BackgroundThemeFactory getFactory(String schemeName) { return factories.get(schemeName); }
}
```

```java
    @FXML
    public void applyBackgroundColor(ActionEvent event) {
        Scene scene = ((Node) event.getSource()).getScene();
        Parent root = scene.getRoot();
        String selectedScheme = strategyComboBox.getValue();

        BackgroundThemeFactory factory = ThemeManager.getFactory(selectedScheme);

        if (factory != null) {
            BackgroundTheme scheme = factory.createTheme();
            scheme.applyBackground(root);
        }
    }
```

sample.fxml

```
<ComboBox onAction="#applyBackgroundColor" fx:id="strategyComboBox" promptText="Background">
    <items>
        <FXCollections fx:factory="observableArrayList">
            <String fx:value="dark" />
            <String fx:value="light" />
            <String fx:value="highcontrast" />
        </FXCollections>
    </items>
</ComboBox>
```

Висновок: у цій лабораторній роботі я ознайомилась з різними шаблонами проєктування і реалізувала шаблон “Factory Method” .