Івано-Франківський національний технічний

університет нафти і газу

Кафедра

інженерії програмного забезпечення

Лабораторна робота №5

Поведінкові шаблони

Виконав

Ст. гр. ІП-22-1

Хімій Денис

Перевірила

Піх М.М.

Івано-Франківськ

2024

Мета: продемонструвати реалізацію поведінкових шаблонів проєктування в коді проєкту

Хід виконання роботи

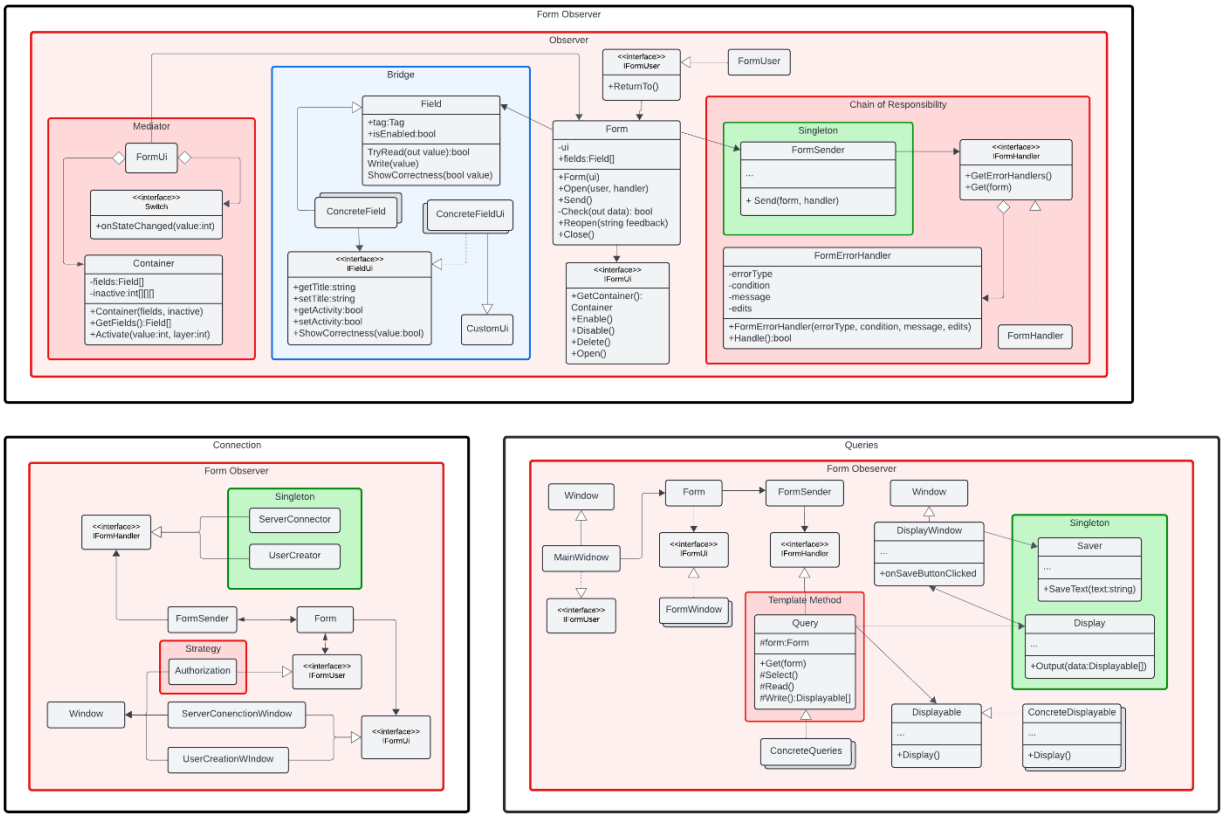

Рис. 1 – Загальна діаграма класів проєкту

Як випливає з рисунку 1 в роботі використано цілих 5 поведінкові шаблони проєктування (позначено червоним): Спостерігач, Шаблонний метод, Ланцюг відповідальності, Стратегія та Посередник. Нижче наведено всі класи, які їх реалізують.

Ланцюг відповідальності

General/Form/Logic/FormSender

```
namespace CS.General.Form.Logic
{
	public static class FormSender
	{
	 public static void Send(Form form, IFormHandler handler)
	 {
	     try { handler.Get(form); }
	     catch (Exception exception)
	     {
	         foreach (var errorHandler in handler.ErrorHandlers)
	             if (errorHandler.Handle(exception, form)) return;
	             }

	             form.Close();
	      }
	}
}
```

General/Form/Logic/FormErrorHandler

```
using CS.General.Form.Field.Logic;

namespace CS.General.Form.Logic
{
	public class FormErrorHandler(Type type, Func<Exception, bool> condition,
Func<Exception, string> message, params (Tag Tag, object Value)[] edits)
	{
	      private Type _type = type;
	 private Func<Exception, bool> _condition = condition;
	 private Func<Exception, string> _message = message;
	 private (Tag Tag, object Values)[] _edits = edits;

	 public bool Handle(Exception exception, Form form)
	 {
	             if ((exception.GetType().Equals(_type) ||
exception.GetType().IsSubclassOf(_type)) && _condition(exception))
	     {
	          foreach ((Tag tag, object value) in _edits) form[tag] = value;
	                   form.Reopen(_message(exception));
	                return true;
	             }

	     return false;
	 }
```

```
            public FormErrorHandler(Func<Exception, string> message, params (Tag Tag, object Value)[] edits) : this(typeof(object), ex => true, message, edits) { }
            public FormErrorHandler(Type type, Func<Exception, bool> condition, string message, params (Tag Tag, object Value)[] edits) : this(type, condition, ex => message, edits) { }
            public FormErrorHandler(Type type, Func<Exception, string> message, params (Tag Tag, object Value)[] edits) : this(type, ex => true, message, edits) { }

      }
}
```

General/Form/IFormHandler

```
namespace CS.General.Form.Logic
{
      public interface IFormHandler
    {
            public FormErrorHandler[] ErrorHandlers { get; }

        public void Get(Form form);
    }
}
```

Посередник

General/Form/UI/Container

```
using Input = CS.General.Form.Field.Logic;

namespace CS.General.Form.Container
{
      public class Container
      {
            private int[][][] _inactive;
            private Input.Field[] _fields;

            public Container(Input.Field[] fields, int[][][] inactive)
            {
                  _fields = fields;
                  _inactive = inactive;
                  if (inactive.Length > 0)
                  {
                        for (int i = 0; i < inactive.Length; i++)
                              Activate(-1, i);
                  }
            }

            public Container(Input.Field[] fields, int[][] inactive) : this(fields, [inactive]) { }

            public Container(Input.Field[] fields) : this(fields, Array.Empty<int[][]>()) { }
```

```
		public Input.Field[] GetFields() => _fields.Where(e => e.IsEnabled).ToArray();

		public void Activate(int value, int layer = 0)
		{
			value = value > _inactive[layer].Length - 1 ? _inactive[layer].Length -
1 : value;
			for (int i = 0; i < _fields.Length; i++)
			{
				if (_inactive[layer].Any(a => a.Contains(i)))
					_fields[i].IsEnabled = !(value == -1 ||
_inactive[layer][value].Contains(i));
			}
		}
	}
}
```

Спостерігач

General/Form/Logic/Form

```
using System.Windows;
using Input = CS.General.Form.Field.Logic;

namespace CS.General.Form.Logic
{
	public class Form
	{
		public object this[Input.Tag tag]
		{
			get => _data[tag];
			set => _sets[tag].Write(value);
		}

		private Dictionary<Input.Tag, object> _data;
		private Dictionary<Input.Tag, Input.Field> _sets;
		private IFormUi _ui;
		private IFormUser _user;
		private IFormHandler _handler;

		public Form(IFormUi ui)
		{
			_ui = ui;
		}

		public void Open(IFormUser user, IFormHandler handler)
		{
			_ui.Open(Send, Close);
			_user = user;
			_handler = handler;
			_user.Hide();
		}

		private void Send()
		{
			_sets = _ui.Container.GetFields().ToDictionary(item => item.Tag, item =>
item);
```

```
                if (Check(out Dictionary<Input.Tag, object> data))
                {
                        _data = data;
                        _ui.Hide();
                        FormSender.Send(this, _handler);
                }
        }

        private bool Check(out Dictionary<Input.Tag, object> data)
        {
                data = [];
                bool result = true;

                foreach (var set in _sets)
                {
                        if (set.Value.TryRead(out object value)) data[set.Key] = value;
                        else result = false;
                }

                return result;
        }

        public void Reopen(string feedback)
        {
                _ui.Show();
                MessageBox.Show(feedback);
        }

        public void Close()
        {
                _user.ReturnTo();
                _ui.Close();
        }
    }
}
```

General/Form/UI/FormButtons.xaml

```
<UserControl x:Class="CS.General.Form.FormButtons"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="45" d:DesignWidth="300" Margin="15 10">

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition Width="150"/>
            <ColumnDefinition Width="150"/>
        </Grid.ColumnDefinitions>

        <Button x:Name="Submit" Content="Підтвердити" Grid.Column="1"
Style="{StaticResource DefaultButton}"/>
        <Button x:Name="Cancel" Content="Скасувати" Grid.Column="2" Style="{StaticResource
DefaultButton}"/>
    </Grid>
</UserControl>
```

General/Form/UI /FormButtons.xaml.cs

```
using System.Windows;
using System.Windows.Controls;
```

```
namespace CS.General.Form
{
	/// <summary>
	/// Interaction logic for FormButtons.xaml
	/// </summary>
	public partial class FormButtons : UserControl
	{
		public FormButtons()
		{
			InitializeComponent();
		}

		public void Connect(Action send, Action close)
		{
			Submit.Click += (object sender, RoutedEventArgs e) => send();
			Cancel.Click += (object sender, RoutedEventArgs e) => close();
		}
	}
}
```

General/Form/Field/Logic/EnumField.cs

```
using CS.General.Form.Field.UI;

namespace CS.General.Form.Field.Logic
{
	internal class EnumField : Field
	{
		public override bool IsEnabled
		{
			get => Ui.IsEnabled;
			set
			{
				if (!value) Ui.Selected = -1;
				Ui.IsEnabled = value;
			}
		}

		protected ListField Ui { get; private set; }

		public EnumField(ListField ui, ComboList type, Tag tag) : base(tag)
		{
			Ui = ui;
			Ui.List = type;
		}

		protected virtual bool Check() => Ui.Selected > -1;

		public override bool TryRead(out object value)
		{
			value = Ui.Selected;
			bool result = Check();
			ShowCorrectness(result);
			return result;
		}

		public override void Write(object value)
		{
			Ui.Selected = (int)value;
			ShowCorrectness(false);
		}

		protected override void ShowCorrectness(bool correct)
		{
			Ui.ShowCorrectness(correct);
		}
```

```
      }
}
```

General/Form/Field/Logic/Field.cs

```
using CS.General.Form.Field.UI;

namespace CS.General.Form.Field.Logic
{
    public abstract class Field(Tag tag)
    {
        public Tag Tag => tag;
             public abstract bool IsEnabled { get; set; }

             public abstract bool TryRead(out object value);

        public abstract void Write(object value);

        protected abstract void ShowCorrectness(bool correct);
       }

    public enum Tag
    {
        Default, Host, Port, Database, Name, Password, Type,
             Unit, ProductCategory, Product, Workshop, Site,
        FirstDate, LastDate,
             EmployeeCategory, Engineer, Laborer,
        Laboratory, ProductsPart
       }
}
```

General/Form/Field/Logic/IntegerField.cs

```
namespace CS.General.Form.Field.Logic
{
    internal class IntegerField(TextField ui, Tag tag) : StringField(ui, tag)
    {
        public override bool TryRead(out object value)
        {
            bool result = base.TryRead(out value) & int.TryParse(value.ToString(), out int
number);
            value = number;
            ShowCorrectness(result);
                   return result;
        }
    }
}
1. General/Form/Field/Logic/OptionalEnumField.cs

namespace CS.General.Form.Field.Logic
{
       internal class OptionalEnumField : EnumField
       {
             public OptionalEnumField(ListField ui, ComboList type, Tag tag) : base(ui, type,
tag)
             {
                   ui.Title += " *";
             }

             protected override bool Check() => true;
       }
}
```

General/Form/Field/Logic/StringField.cs

```
using CS.General.Form.Field.UI;

namespace CS.General.Form.Field.Logic
{
	public class StringField(ITextFieldUi ui, Tag tag) : Field(tag)
    {
		public override bool IsEnabled
		{
			get => Ui.IsEnabled;
			set
			{
				if (!value) Ui.Text = string.Empty;
				Ui.IsEnabled = value;
			}
		}

		protected ITextFieldUi Ui { get; private set; } = ui;

		public override bool TryRead(out object value)
        {
            value = Ui.Text;
			bool result = Ui.Text.Length > 0;
			ShowCorrectness(result);
			return result;
        }

		public override void Write(object value)
		{
			Ui.Text = value.ToString();
			ShowCorrectness(false);
		}

		protected override void ShowCorrectness(bool correct)
		{
			Ui.ShowCorrectness(correct);
		}
	}
}
```

General/Form/Field/UI/DateField.xaml

```
<UserControl x:Class="CS.General.Form.Field.DateField"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="450" d:DesignWidth="800" Margin="20 10">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="20"/>
            <RowDefinition Height="25"/>
        </Grid.RowDefinitions>

        <TextBlock Name="Header" VerticalAlignment="Center"/>
        <DatePicker Name="Field" Grid.Row="1" VerticalAlignment="Center"/>
    </Grid>
</UserControl>
```

General/Form/Field/UI/DateField.xaml.cs

```
using CS.General.Form.Field.UI;
```

```
using System.Windows.Controls;
using System.Windows.Controls.Primitives;
using System.Windows.Media;

namespace CS.General.Form.Field
{
	public partial class DateField : UserControl, ITextFieldUi
	{
		public string Title
		{
			get => Header.Text;
			set => Header.Text = value;
		}

		public string Text
		{
			get		=>		Field.SelectedDate.HasValue		?
Field.SelectedDate.Value.ToString("yyyy-MM-dd") : string.Empty;
			set  =>  Field.SelectedDate  =  DateTime.ParseExact(value,  "yyyy-MM-dd",
null);
		}

		public DateField()
		{
			InitializeComponent();

			Field.Loaded += (s, e) =>
			{
				if	(Field.Template.FindName("PART_TextBox",	Field)	is
DatePickerTextBox datePickerTextBox)
				{
					var		watermarkProperty		=
typeof(DatePickerTextBox).GetProperty("Watermark",  System.Reflection.BindingFlags.NonPublic
| System.Reflection.BindingFlags.Instance);
					watermarkProperty?.SetValue(datePickerTextBox,	"Оберіть
дату");
				}
			};
		}

		public void ShowCorrectness(bool correct)
		{
			Header.Foreground = correct ? Brushes.Black : Brushes.Red;
		}
	}
}
```

General/Form/Field/UI/IFieldUi.cs

```
namespace CS.General.Form.Field.UI
{
	public interface IFieldUi
	{
		public string Title { get; set; }
		public bool IsEnabled { get; set; }

		public void ShowCorrectness(bool correct);
	}
}
2. General/Form/Field/UI/ITextFieldUi.cs

namespace CS.General.Form.Field.UI
{
	public interface ITextFieldUi : IFieldUi
	{
		public string Text { get; set; }
```

```
      }
}
```

3. General/Form/Field/UI/ListField.xaml

```
<UserControl x:Class="CS.General.Form.Field.ListField"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="45" d:DesignWidth="800" Margin="20 10">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="20"/>
            <RowDefinition Height="25"/>
        </Grid.RowDefinitions>

        <TextBlock Name="Header" Grid.ColumnSpan="2"/>
        <ComboBox  Name="Combo"  Grid.Row="1"  Padding="2.5  4.5"  VerticalAlignment="Center"
DropDownClosed="Combo_DropDownClosed"/>
    </Grid>
</UserControl>
```

General/Form/Field/UI/ListField.xaml.cs

```
using CS.General.Form.Field.UI;
using System.Windows.Controls;
using System.Windows.Media;

namespace CS.General.Form.Field
{
       public partial class ListField : UserControl, IFieldUi
       {
              public int Selected
              {
                     get => Combo.SelectedIndex;
                     set => Combo.SelectedIndex = value;
              }

              public string Title
              {
                     get => Header.Text;
                     set => Header.Text = value;
              }

              private int _previousSelected = -1;

              public ComboList List { set => Combo.ItemsSource = Database.ComboLists[value];
}

              public void SetEvent(Action<int> action)
              {
                     Combo.SelectionChanged += (s, e) => action(Selected);
              }

              public ListField()
              {
                     InitializeComponent();
              }

              public void ShowCorrectness(bool correct)
              {
                     Header.Foreground = correct ? Brushes.Black : Brushes.Red;
              }

              private void Combo_DropDownClosed(object sender, EventArgs e)
              {
                     if (_previousSelected == Selected)
                     {
```

```
                        Selected = -1;
                        _previousSelected = -1;
                    }
                    else _previousSelected = Selected;
                }
        }
}
```

General/Form/Field/UI/TextField.xaml

```
<UserControl x:Class="CS.General.Form.Field.TextField"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="45" d:DesignWidth="800" Margin="20 10">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="20"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <TextBlock Name="Header" Grid.ColumnSpan="2"/>
        <TextBox Name="Field" Grid.Row="1" Padding="2.5 4.5" VerticalAlignment="Center"/>
    </Grid>
</UserControl>
```

General/Form/Field/UI/TextField.xaml.cs

```
using CS.General.Form.Field.UI;
using System.Windows.Controls;
using System.Windows.Media;

namespace CS.General.Form.Field
{
        public partial class TextField : UserControl, ITextFieldUi
        {
                public string Title
                {
                        get => Header.Text;
                        set => Header.Text = value;
                }

                public string Text
                {
                        get => Field.Text;
                        set => Field.Text = value;
                }

                public void ShowCorrectness(bool correct)
                {
                        Header.Foreground = correct ? Brushes.Black : Brushes.Red;
                }

                public TextField()
                {
                        InitializeComponent();
                }
        }
}
```

Стратегія

Authorization/Window

```
using CS.Authorization;
using CS.General;
using CS.General.Form.Logic;
using System.Windows;

namespace CS
{
	public partial class MainWindow : Window, IFormUser
	{
		private static Dictionary<ConnectionStatus, bool[]> _statuses = [];

		public void SetStatus()
		{
			Connection.IsEnabled = _statuses[Database.Status][0];
			Creating.IsEnabled = _statuses[Database.Status][1];
			Editing.IsEnabled = _statuses[Database.Status][2];
		}

		public MainWindow()
		{
			InitializeComponent();
			SetStatus();
		}

		static MainWindow()
		{
			_statuses[ConnectionStatus.None] = [true, false, false];
			_statuses[ConnectionStatus.Connected] = [true, true, true];
		}

		private void Editing_Click(object sender, RoutedEventArgs e)
		{
			new Queries.MainWindow().Show();
			Close();
		}

		private void Creating_Click(object sender, RoutedEventArgs e)
		{
			new Form(new UserCreatorWindow()).Open(this, UserCreator.Instant());
			Hide();
		}

		private void Connection_Click(object sender, RoutedEventArgs e)
		{
			new Form(new ServerConnectorWindow()).Open(this, ServerConnector.Instant());
			Hide();
		}

		public void ReturnTo()
		{
			Show();
			SetStatus();
		}
	}

	public enum ConnectionStatus
	{
		None, Connected
	}
}
```

Шаблонний метод

Queries/Query

```
using CS.General;
using CS.General.Form.Field.Logic;
using CS.General.Form.Logic;
using CS.Output;
using CS.Output.Items;
using Npgsql;

namespace CS.Queries
{
	public abstract class Query : IFormHandler
	{
		private string _select = string.Empty;

		public FormErrorHandler[] ErrorHandlers => [
			new FormErrorHandler(typeof(PostgresException), (e) => $"Помилка роботи з
PostgreSQL\n\n{_select}\n\n{e.Message}"),
			new FormErrorHandler((e) => $"Необроблена помилка ({e.Message})"),
		];

		protected List<Displayable> Result { get; } = [];
		protected Form Form { get; private set; }

		public void Get(Form form)
		{
			Form = form;
			_select = Select();
			using (var reader = new NpgsqlCommand(_select,
Database.Connection).ExecuteReader())
			{
				while (reader.Read()) Read(reader);
			}
			Display.Output(Write());
		}

		protected abstract string Select();

		protected abstract void Read(NpgsqlDataReader reader);

		protected virtual List<Displayable> Write() => Result;

		protected string CheckOptional(Tag tag, string name, int addition = 1) => (int)Form[tag]
== -1 ? "true" : $"{name} = {(int)Form[tag] + addition}";

		protected string CheckSwitch(Tag tag, (Tag tag, string name, int addition)[] options)
		{
			int selected = (int)Form[tag] - 1;
			return selected == -1 ? "true" : $"{options[selected].name} =
{(int)Form[options[selected].tag] + options[selected].addition}";
		}
	}
}
```

Висновок

На цій лабораторній роботі продемонструвати реалізацію поведінкових шаблонів проєктування в коді проєкту.