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 №4

Технології розробки програмного забезпечення

«Вступ до патернів проектування»

«Веб-браузер»

Виконав:
студент групи ІА-33
Мартинюк Ю.Р.

Перевірив:
Мягкий Михайло Юрійович

Київ 2025

Тема: Вступ до патернів проектування

Мета: Вивчити структуру шаблонів «Singleton», «Iterator», «Proxy», «State», «Strategy» та навчитися застосовувати їх в реалізації програмної системи

Зміст

Завдання

• Ознайомитись з короткими теоретичними відомостями.

• Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.

• Реалізувати один з розглянутих шаблонів за обраною темою.

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

• Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму класів, яка представляє використання шаблону в реалізації системи, навести фрагменти коду по реалізації цього шаблону.

Теоретичні відомості

Будь-який патерн проєктування, використовуваний при розробці інформаційних систем, являє собою формалізований опис, який часто зустрічається в завданнях проєктування, вдале рішення даної задачі, а також рекомендації по застосуванню цього рішення в різних ситуаціях. Крім того, патерн проєктування обов’язково має загальновживане найменування. Правильно сформульований патерн проєктування дозволяє, відшукавши одного разу вдале

рішення, користуватися ним знову і знову. Варто підкреслити, що важливим початковим етапом при роботі з патернами є адекватне моделювання розглянутої предметної області. Це є необхідним як для отримання належним чином формалізованої постановки задачі, так і для вибору відповідних патернів проєктування.

Відповідне використання патернів проєктування дає розробнику ряд незаперечних переваг. Наведемо деякі з них. Модель системи, побудована в межах патернів проєктування, фактично є структурованим виокремленням тих елементів і зв'язків, які значимі при вирішенні поставленого завдання. Крім цього, модель, побудована з використанням патернів проєктування, більш проста і наочна у вивченні, ніж стандартна модель. Проте, не дивлячись на простоту і наочність, вона дозволяє глибоко і всебічно опрацювати архітектуру розроблюваної системи з використанням спеціальної мови.

Застосування патернів проєктування підвищує стійкість системи до зміни вимог та спрощує неминуче подальше доопрацювання системи. Крім того, важко переоцінити роль використання патернів при інтеграції інформаційних систем організації. Також слід зазначити, що сукупність патернів проєктування, по суті, являє собою єдиний словник проєктування, який, будучи уніфікованим засобом, незамінний для спілкування розробників один одним.

Таким чином шаблони представляють собою, підтверджені роками розробок в різних компаніях і на різних проєктах, «ескізи» архітектурних рішень, які зручно застосовувати у відповідних обставинах.

Тема проєкту

6. Web-browser (proxy, chain of responsibility, factory method, template method, visitor, p2p) Веб-браузер повинен мати можливість зробити наступне: мати адресний рядок для введення адреси сайту, переміщатися і відображати структуру html документа, переглядати підключений javascript та css файли, перегляд всіх підключених ресурсів (зображень), коректна обробка відповідей з сервера (коди відповідей HTTP) – переходи при перенаправленнях, відображення сторінок 404 і 502/503.

Посилання на Github: https://github.com/masonabor/web-browser-lab/tree/main/lab4

Хід роботи

Діаграма класів

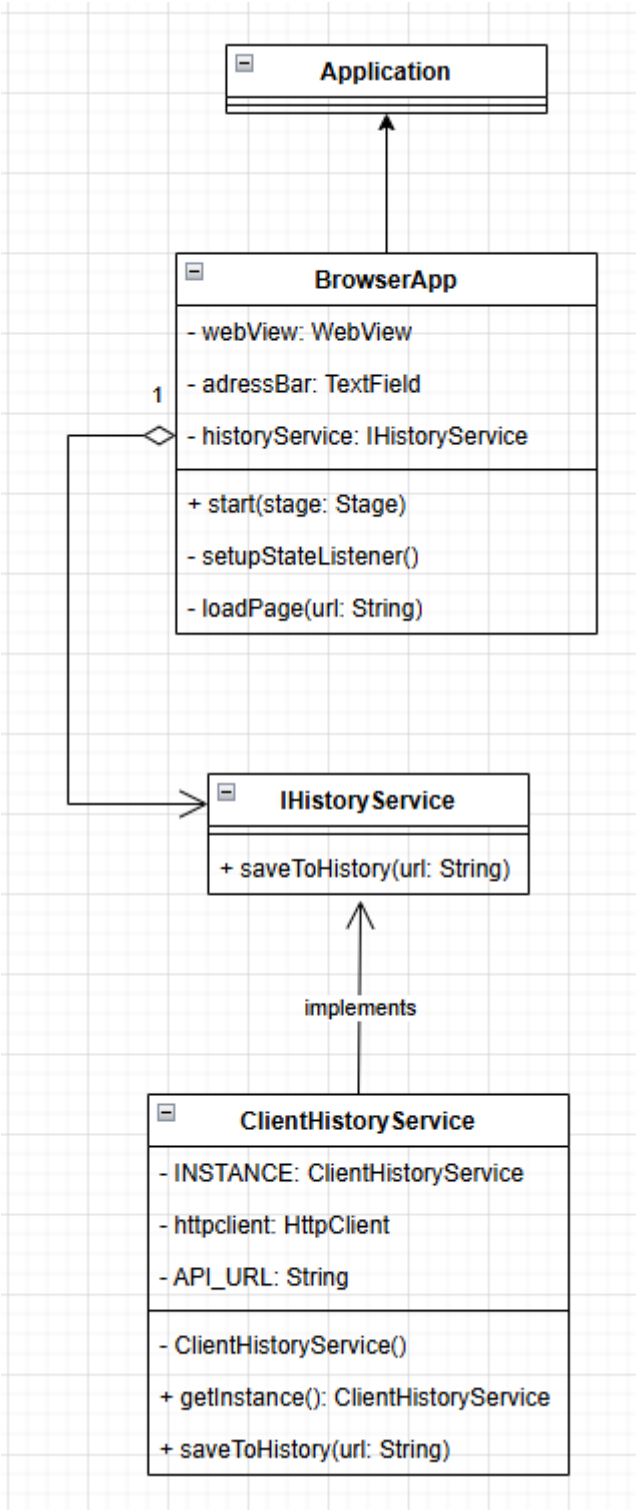

Рис.1 – Діаграма компонентів для BrowserApp (клієнтська частина)

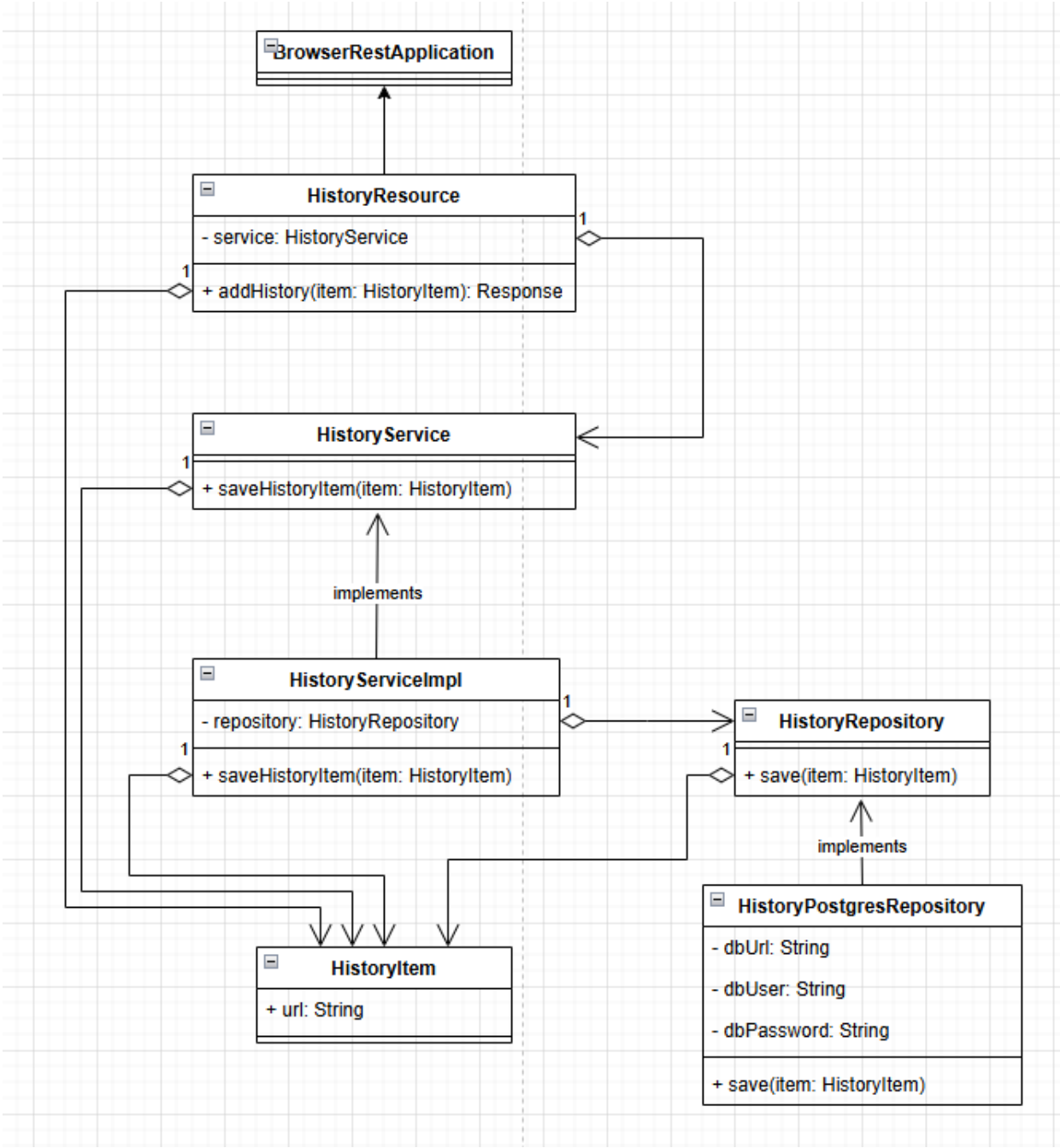

Рис.2 – Діаграма класів для Rest-Api-Browser (серверна частина)

Обгрунтування використання патернів проєктування

1. Singleton
   - Де використано: ClientHistoryService (на стороні JavaFX-клієнта).
   - Цей патерн гарантує, що у всього клієнтського додатку існує лише один екземпляр сервісу для роботи з історією. Це критично важливо для ефективного управління ресурсами. Замість того, щоб створювати новий HttpClient (який керує пулом з'єднань) кожного разу, коли нам потрібно щось зберегти, ми створюємо один сервіс, який живе протягом усього

життєвого циклу програми. Це забезпечує єдину точку доступу до сервісу історії з будь-якої частини додатку.

2. Proxy
- Де використано: ClientHistoryService (на стороні JavaFX-клієнта).
- Цей патерн ідеально підходить для приховування складності. Основний клас BrowserApp не повинен знати, *як* спілкуватися з сервером. Він не повинен формувати JSON, налаштовувати HTTP-заголовки, відправляти асинхронні запити чи обробляти відповіді. ClientHistoryService виступає як "замісник" (proxy) реального сервісу, що знаходиться в мережі. BrowserApp просто викликає локальний метод saveToHistory(url), а вся складна мережева логіка "захована" всередині проксі. Це робить код BrowserApp чистішим і спрощує його підтримку.

3. State
- Де використано: У BrowserApp, при прослуховуванні стану LoadWorker у WebView.
- Нам потрібно виконати дію (зберегти історію) лише тоді, коли WebView переходить у конкретний стан — Worker.State.SUCCEEDED (успішне завантаження). Патерн State дозволяє об'єкту (в нашому випадку LoadWorker) змінювати свою поведінку залежно від свого внутрішнього стану. Ми підписуємося на ці зміни стану і реагуємо лише на той, який нас цікавить. Це набагато чистіший підхід, ніж постійно перевіряти статус завантажувача в циклі.

Код програми

BrowserApp.java

```
package com.edu.web.browserapp;

import com.edu.web.browserapp.service.IHistoryService;
import com.edu.web.browserapp.service.impl.ClientHistoryService;
import javafx.application.Application;
import javafx.concurrent.Worker;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.input.KeyCode;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
```

```
import javafx.scene.web.WebView;
import javafx.stage.Stage;

public class BrowserApp extends Application {

    private WebView webView;
    private TextField addressBar;
    private Label statusLabel;
    private IHistoryService historyService;

    @Override
    public void start(Stage stage) {
        historyService = ClientHistoryService.getInstance();

        webView = new WebView();
        addressBar = new TextField("https://www.google.com");
        statusLabel = new Label("Готовий");

        addressBar.setOnKeyPressed(e -> {
            if (e.getCode() == KeyCode.ENTER) {
                loadPage(addressBar.getText());
            }
        });


        webView.getEngine().getLoadWorker().stateProperty().addListener((obs,
oldState, newState) -> {
            statusLabel.setText(newState.toString());

            if (newState == Worker.State.SUCCEEDED) {
                String loadedUrl = webView.getEngine().getLocation();
                statusLabel.setText("Завантажено: " + loadedUrl);

                historyService.saveToHistory(loadedUrl);
            }
        });

        HBox topBar = new HBox(10, new Label("URL:"), addressBar);
        topBar.setPadding(new Insets(10));
        addressBar.prefWidthProperty().bind(topBar.widthProperty().subtract(50));
```

```
        BorderPane root = new BorderPane();
        root.setTop(topBar);
        root.setCenter(webView);
        root.setBottom(statusLabel);
        BorderPane.setMargin(statusLabel, new Insets(5));

        loadPage(addressBar.getText());

        Scene scene = new Scene(root, 1024, 768);
        stage.setTitle("Web Browser");
        stage.setScene(scene);
        stage.show();
    }

    private void loadPage(String url) {
        if (!url.startsWith("http://") && !url.startsWith("https://")) {
            url = "https://" + url;
        }
        webView.getEngine().load(url);
    }
}
```

Launcher.java

```
package com.edu.web.browserapp;

import javafx.application.Application;

public class Launcher {
    public static void main(String[] args) {
        Application.launch(BrowserApp.class, args);
    }
}
```

IHistoryService.java

```
package com.edu.web.browserapp.service;

public interface IHistoryService {
    void saveToHistory(String url);
}
```

ClientHistoryService.java

```
package com.edu.web.browserapp.service.impl;

import com.edu.web.browserapp.service.IHistoryService;

import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class ClientHistoryService implements IHistoryService {

   private static final ClientHistoryService INSTANCE = new ClientHistoryService();

   private final HttpClient httpClient;
   private final String API_URL = "http://localhost:8080/browser-service-
1.0/api/history"; // Оновіть назву .war

   private ClientHistoryService() {
      this.httpClient = HttpClient.newBuilder()
            .version(HttpClient.Version.HTTP_1_1)
            .build();
   }

   public static ClientHistoryService getInstance() {
      return INSTANCE;
   }

   @Override
   public void saveToHistory(String url) {
      try {
         String jsonPayload = "{\"url\":\"" + url.replace("\"", "\\\"") + "\"}";

         HttpRequest request = HttpRequest.newBuilder()
               .uri(URI.create(API_URL))
               .header("Content-Type", "application/json")
               .POST(HttpRequest.BodyPublishers.ofString(jsonPayload))
               .build();
```

```
            httpClient.sendAsync(request, HttpResponse.BodyHandlers.ofString())
                    .thenAccept(this::handleResponse)
                    .exceptionally(this::handleError);

        } catch (Exception e) {
            System.err.println("[Proxy] Error during request: " + e.getMessage());
        }
    }

    private void handleResponse(HttpResponse<String> response) {
        if (response.statusCode() == 201) {
            System.out.println("[Proxy] History successfully saved");
        } else {
            System.err.println("[Proxy] Server error" + response.statusCode());
        }
    }

    private Void handleError(Throwable ex) {
        System.err.println("[Proxy] Error during connecting to History Service: " +
ex.getMessage());
        return null;
    }

}
```

BrowserRestApplication.java

```
package com.edu.web.restservicewebbrowser;

import jakarta.ws.rs.ApplicationPath;
import jakarta.ws.rs.core.Application;

@ApplicationPath("/api")
public class BrowserRestApplication extends Application {
}
```

HistpryRepository.java

```
package com.edu.web.restservicewebbrowser.repository;
```

```
import com.edu.web.restservicewebbrowser.domain.HistoryItem;

import java.sql.SQLException;

public interface HistoryRepository {
   void save(HistoryItem HistoryItem) throws SQLException;
}
```

HistoryPostgresRepository.java

```
package com.edu.web.restservicewebbrowser.repository.impl;

import com.edu.web.restservicewebbrowser.domain.HistoryItem;
import com.edu.web.restservicewebbrowser.repository.HistoryRepository;
import jakarta.enterprise.context.ApplicationScoped;
import jakarta.inject.Inject;
import org.eclipse.microprofile.config.inject.ConfigProperty; // Імпорт з MicroProfile

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.time.LocalDateTime;

@ApplicationScoped
public class HistoryPostgresRepository implements HistoryRepository {

   @Inject
   @ConfigProperty(name = "DB_URL")
   private String dbUrl;

   @Inject
   @ConfigProperty(name = "DB_USER")
   private String dbUser;

   @Inject
   @ConfigProperty(name = "DB_PASSWORD")
   private String dbPassword;
```

```
    public void save(HistoryItem item) throws SQLException {
        String sql = "INSERT INTO HistoryItem (url, visit_time) VALUES (?, ?)";

        try (Connection conn = DriverManager.getConnection(dbUrl, dbUser,
dbPassword);
             PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setString(1, item.url());
            pstmt.setTimestamp(2, Timestamp.valueOf(LocalDateTime.now()));
            pstmt.executeUpdate();
        }
    }
}
```

HistoryService.java

```
package com.edu.web.restservicewebbrowser.service;

import com.edu.web.restservicewebbrowser.domain.HistoryItem;

import java.sql.SQLException;

public interface HistoryService {
    void saveHistoryItem(HistoryItem historyItem) throws SQLException;
}
```

HistoryServiceImpl.java

```
package com.edu.web.restservicewebbrowser.service.impl;

import com.edu.web.restservicewebbrowser.domain.HistoryItem;
import com.edu.web.restservicewebbrowser.repository.HistoryRepository;
import com.edu.web.restservicewebbrowser.service.HistoryService;
import jakarta.enterprise.context.ApplicationScoped;
import jakarta.inject.Inject;

import java.net.URI;
import java.net.URISyntaxException;
import java.sql.SQLException;
import java.util.Set;

@ApplicationScoped
```

```
public class HistoryServiceImpl implements HistoryService {

    private static final Set<String> IGNORED_PROTOCOLS = Set.of(
            "about",
            "chrome",
            "file",
            "data"
    );

    private static final int MAX_URL_LENGTH = 2048;

    @Inject
    private HistoryRepository repository;

    public void saveHistoryItem(HistoryItem item) throws SQLException {

        if (item == null || item.url() == null || item.url().isBlank()) {
            throw new IllegalArgumentException("History item або URL не може бути
порожнім.");
        }

        String urlString = item.url().trim();
        URI uri;

        try {
            uri = new URI(urlString);

            if (!uri.isAbsolute()) {
                throw new URISyntaxException(urlString, "URL повинен мати протокол
(scheme).");
            }

        } catch (URISyntaxException e) {
            throw new IllegalArgumentException("Некоректний формат URL: " +
e.getMessage(), e);
        }

        String protocol = uri.getScheme();

        if (protocol == null ||
IGNORED_PROTOCOLS.contains(protocol.toLowerCase())) {
```

```
            System.out.println("Ігнорування URL з протоколом: " + protocol);
            return;
        }

        if (urlString.length() > MAX_URL_LENGTH) {
            throw new IllegalArgumentException("URL занадто довгий (макс " +
MAX_URL_LENGTH + " символів).");
        }

        repository.save(item);
    }
}
```

HistoryItem.java

```
package com.edu.web.restservicewebbrowser.domain;

public record HistoryItem(String url) {
}
```

HistoryResource.java

```
package com.edu.web.restservicewebbrowser.api;

import com.edu.web.restservicewebbrowser.domain.HistoryItem;
import com.edu.web.restservicewebbrowser.service.HistoryService;
import jakarta.inject.Inject;
import jakarta.ws.rs.Consumes;
import jakarta.ws.rs.POST;
import jakarta.ws.rs.Path;
import jakarta.ws.rs.Produces;
import jakarta.ws.rs.core.MediaType;
import jakarta.ws.rs.core.Response;

import java.sql.SQLException;

@Path("/history")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public class HistoryResource {

    @Inject
```

```
    private HistoryService service;

    @POST
    public Response addHistory(HistoryItem item) {
        try {
            service.saveHistoryItem(item);
            return Response.status(Response.Status.CREATED).build();
        } catch (SQLException e) {
            e.printStackTrace();
            return Response.status(Response.Status.INTERNAL_SERVER_ERROR)
                    .entity("Помилка збереження")
                    .build();
        }
    }
}
```

Скриншот застосунку

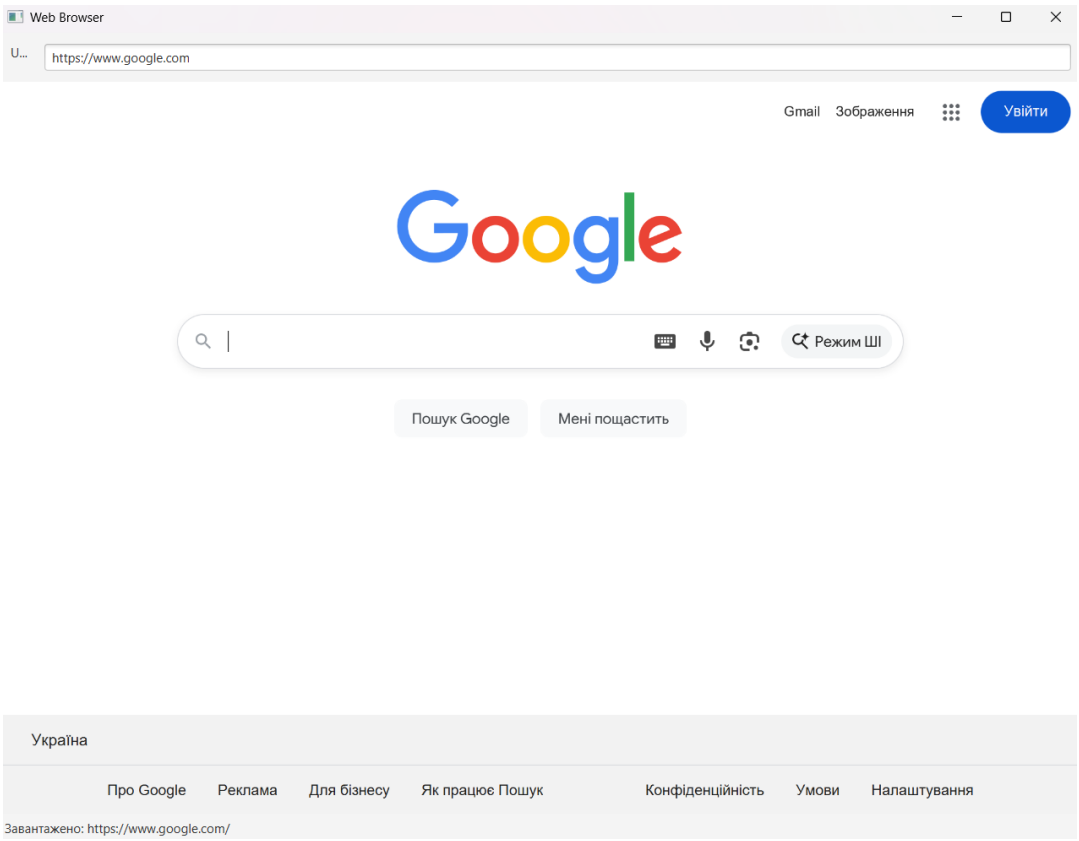

Рис.3 – Скриншот застосунку BrowserApp

Контрольні запитання

1. Що таке шаблон проєктування?
Шаблон проєктування — це перевірене рішення типової задачі, що часто зустрічається під час розробки програмного забезпечення.

2. Навіщо використовувати шаблони проєктування?
Використання шаблонів дозволяє спростити проєктування, зробити код зрозумілішим, зменшити дублювання та полегшити підтримку системи.

3. Яке призначення шаблону «Стратегія»?
Шаблон «Стратегія» призначений для вибору алгоритму під час виконання програми.

4. Нарисуйте структуру шаблону «Стратегія».
Структура включає інтерфейс стратегії, конкретні реалізації стратегій і контекст, який використовує вибрану стратегію.

5. Які класи входять в шаблон «Стратегія», та яка між ними взаємодія?
Інтерфейс описує спільну поведінку, конкретні стратегії реалізують її по-своєму, а контекст делегує виконання вибраній стратегії, не знаючи деталей реалізації.

6. Яке призначення шаблону «Стан»?
Шаблон «Стан» використовується для зміни поведінки об’єкта залежно від його внутрішнього стану.

7. Нарисуйте структуру шаблону «Стан».
Вона складається з інтерфейсу стану, конкретних станів і контексту, який зберігає поточний стан.

8. Які класи входять в шаблон «Стан», та яка між ними взаємодія?
Контекст має посилання на поточний стан, викликає методи стану, а стан може змінювати контекст, передаючи керування іншому стану.

9. Яке призначення шаблону «Ітератор»?
Шаблон «Ітератор» дозволяє послідовно перебирати елементи колекції без розкриття її внутрішньої структури.

10. Нарисуйте структуру шаблону «Ітератор».
Структура включає інтерфейс ітератора, конкретний ітератор і колекцію, яка створює цей ітератор.

11. Які класи входять в шаблон «Ітератор», та яка між ними взаємодія?
Колекція створює ітератор, який надає методи доступу до елементів по одному (наприклад, hasNext, next).

12. В чому полягає ідея шаблона «Одинак»?
Шаблон «Одинак» забезпечує існування лише одного екземпляра класу в межах програми та надає глобальну точку доступу до нього.

13. Чому шаблон «Одинак» вважають «анти-шаблоном»?
Його вважають антипатерном, бо він ускладнює тестування, порушує

інкапсуляцію та створює приховані залежності.

14. Яке призначення шаблону «Проксі»?
Шаблон «Проксі» використовується для контролю доступу до іншого об’єкта або додавання додаткової логіки без зміни його коду.

15. Нарисуйте структуру шаблону «Проксі».
Вона включає інтерфейс, реальний об’єкт і проксі-клас, який реалізує той самий інтерфейс і перехоплює виклики.

16. Які класи входять в шаблон «Проксі», та яка між ними взаємодія?
Інтерфейс визначає спільні методи, реальний об’єкт виконує основну роботу, а проксі-клас викликає його методи та додає додаткову поведінку (наприклад, кешування або перевірку доступу).

Висновки

Під час виконання даної лабораторної роботи я ознайомився з патернами проектування та застосував їх на практиці для власного браузера.