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
З дисципліни: «Теорія розробки програмного забезпечення»
Web-browser
Proxy

Виконав:

студент групи ІА-14
Калінін Я.В

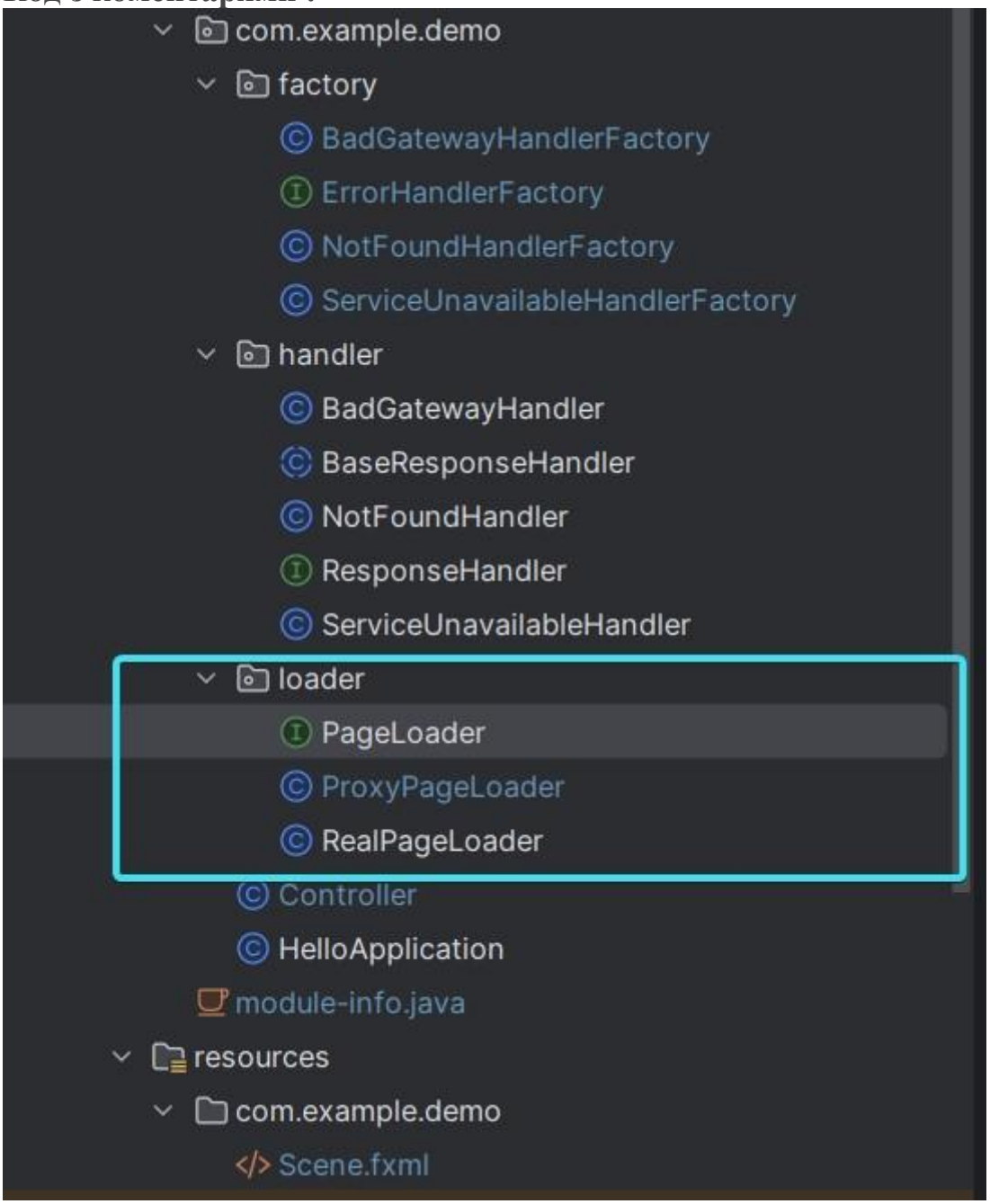
Перевірив:
Мягкий М.Ю

Київ 2023

Мета: Реалізувати шаблон проектування «Proxy» до заданої теми :
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.

Теоритичні відомості :
Замісник — це структурний патерн проектування, що дає змогу підставляти замість реальних об’єктів спеціальні об’єкти-замінники. Ці об’єкти перехоплюють виклики до оригінального об’єкта, дозволяючи зробити щось *до* чи *після* передачі виклику оригіналові. По суті об’єкти замінники будуть контролювати доступ до оригінального об’єкту. Клас дублер реалізує той самий інтерфейс що і оригінальний об’єкт і При отриманні запиту від клієнта об’єкт-замісник сам би створював примірник службового об’єкта та переадресовував би йому всю реальну роботу.

Код з коментарями :



Інтерфейс **«PageLoader»** - Цей інтерфейс створений для того, щоб ізолювати логіку роботи з веб-сторінками від конкретної реалізації, щоб забезпечити гнучкість та можливість розширення.Також його реалізація необіхдна для того щоб об’єкт-замісник можна передати до будь-якого коду, що очікує на сервісний об’єкт.

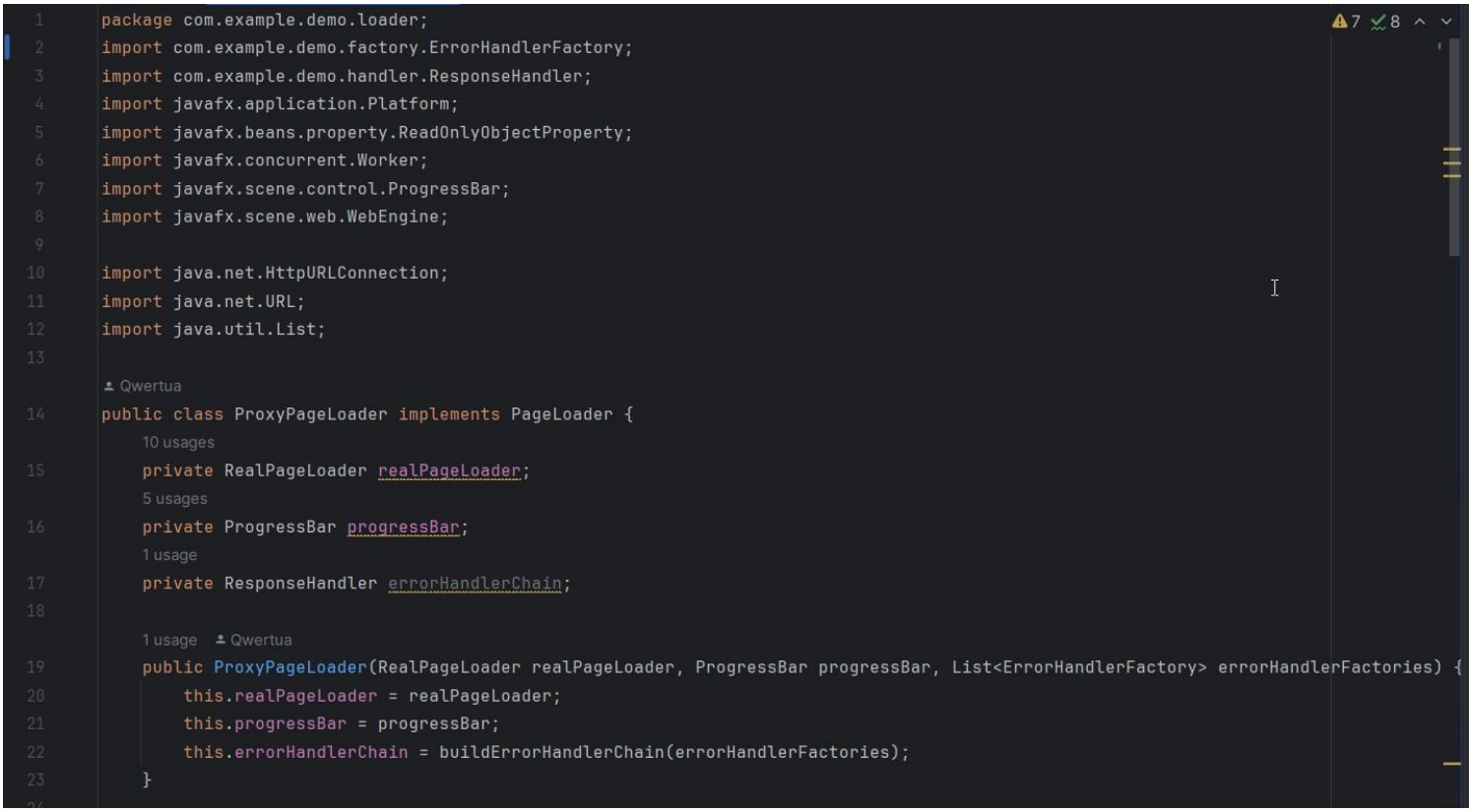
```java
package com.example.demo.loader;
import javafx.beans.property.ReadOnlyObjectProperty;
import javafx.concurrent.Worker;

2 implementations  Qwertua *
public interface PageLoader {
    /* Завантаження сторінки з вказаним URL*/
    2 usages  2 implementations  Qwertua
    void loadPage(String url);
    /* метод для перезавантаження сторінки*/
    2 usages  2 implementations  Qwertua
    void reloadPage();

    /*Цей метод , який відображає стан робочого потоку, використовуваного для завантаження сторінки.
    В нащому випадку дозволяє спостерігати за змінами стану робочого потоку для визначення
    завершення завантаження сторінки.
    */
    4 usages  2 implementations  Qwertua
    ReadOnlyObjectProperty<Worker.State> getLoadWorkerStateProperty();
}
```

class RealPageLoader

```java
package com.example.demo.loader;
import javafx.beans.property.ReadOnlyObjectProperty;
import javafx.concurrent.Worker;
import javafx.scene.web.WebEngine;

Qwertua *
public class RealPageLoader implements PageLoader {
    5 usages
    private WebEngine engine;

    1 usage  Qwertua
    public RealPageLoader(WebEngine engine) {
        this.engine = engine;
    }
    /*Реалізація методу loadPage, який використовує внутрішні методи класу WebEngine
    для завантаження веб-сторінки за вказаним URL.
    */
    2 usages  Qwertua
    @Override
    public void loadPage(String url) {
        engine.load( s: "http://" + url);
    }
    /*Реалізація методу reloadPage, який використовує внутрішні методи класу WebEngine
    для перезавантаження вже завантаженої сторінки.
    */
```

```
    2 usages  Qwertua
    @Override
    public void reloadPage() {
        engine.reload();
    }
    /*Реалізація методу getLoadWorkerStateProperty, який повертає
    властивість стану робочого потоку для відстеження стану завантаження сторінки.
    */

    4 usages  Qwertua
    @Override
    public ReadOnlyObjectProperty<Worker.State> getLoadWorkerStateProperty() {
        return engine.getLoadWorker().stateProperty();
    }
    /*отримання доступу до внутрішніх методів WebEngine*/
    4 usages  Qwertua
    public WebEngine getEngine() {
        return engine;
    }
}
```

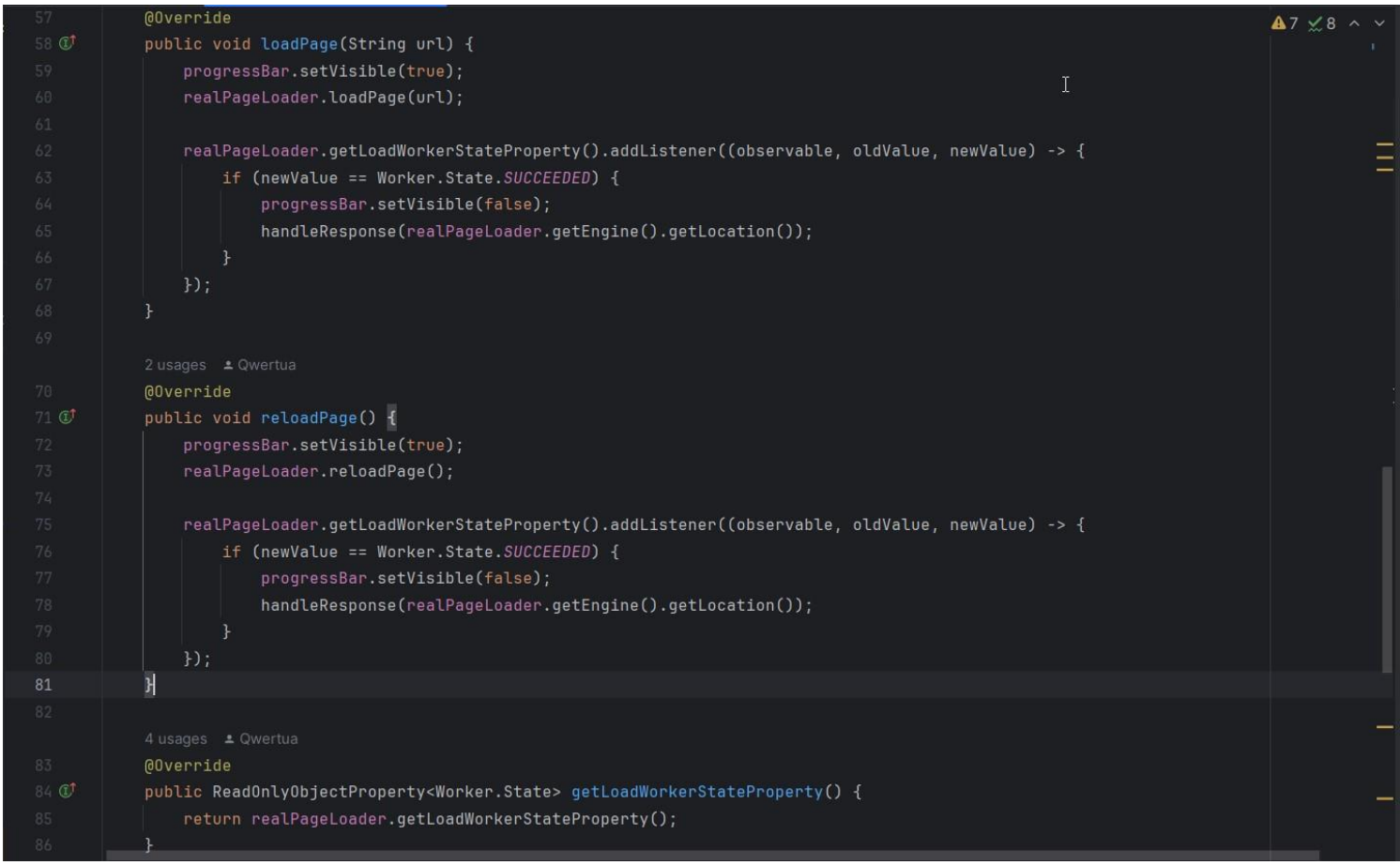
class ProxyPageLoader



```
package com.example.demo.loader;
import com.example.demo.factory.ErrorHandlerFactory;
import com.example.demo.handler.ResponseHandler;
import javafx.application.Platform;
import javafx.beans.property.ReadOnlyObjectProperty;
import javafx.concurrent.Worker;
import javafx.scene.control.ProgressBar;
import javafx.scene.web.WebEngine;

import java.net.HttpURLConnection;
import java.net.URL;
import java.util.List;

 Qwertua
public class ProxyPageLoader implements PageLoader {
    10 usages
    private RealPageLoader realPageLoader;
    5 usages
    private ProgressBar progressBar;
    1 usage
    private ResponseHandler errorHandlerChain;

    1 usage  Qwertua
    public ProxyPageLoader(RealPageLoader realPageLoader, ProgressBar progressBar, List<ErrorHandlerFactory> errorHandlerFactories) {
        this.realPageLoader = realPageLoader;
        this.progressBar = progressBar;
        this.errorHandlerChain = buildErrorHandlerChain(errorHandlerFactories);
    }
```

У конструкторі створюється об'єкт ProxyPageLoader з переданим об'єктом RealPageLoader (реальний завантажувач сторінок), ProgressBar та списком фабрик обробників помилок (ErrorHandlerFactory).

Цей конструктор використовується для налаштування проксі-об'єкта для обробки запитів на завантаження сторінок та відображення прогресу завантаження.

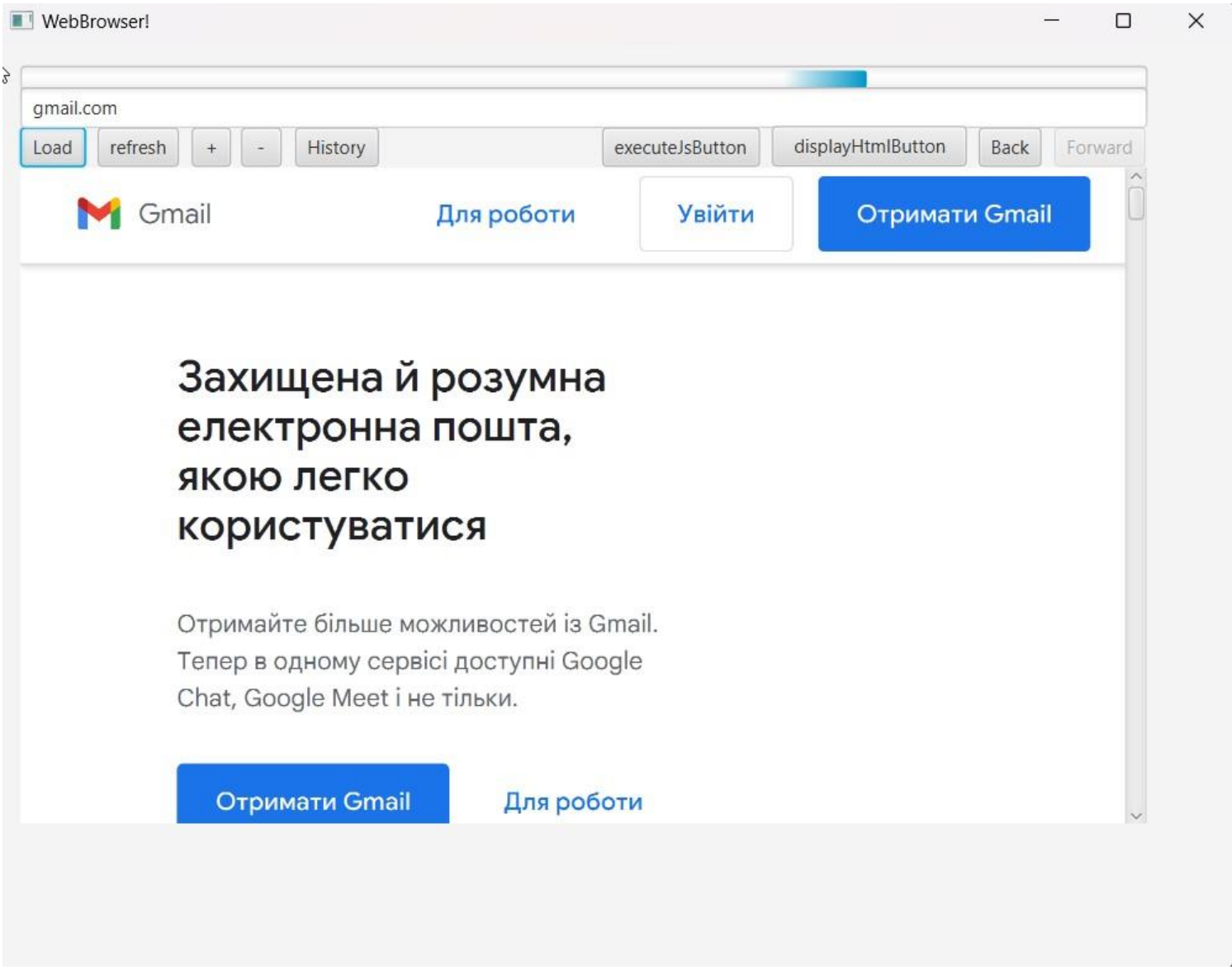


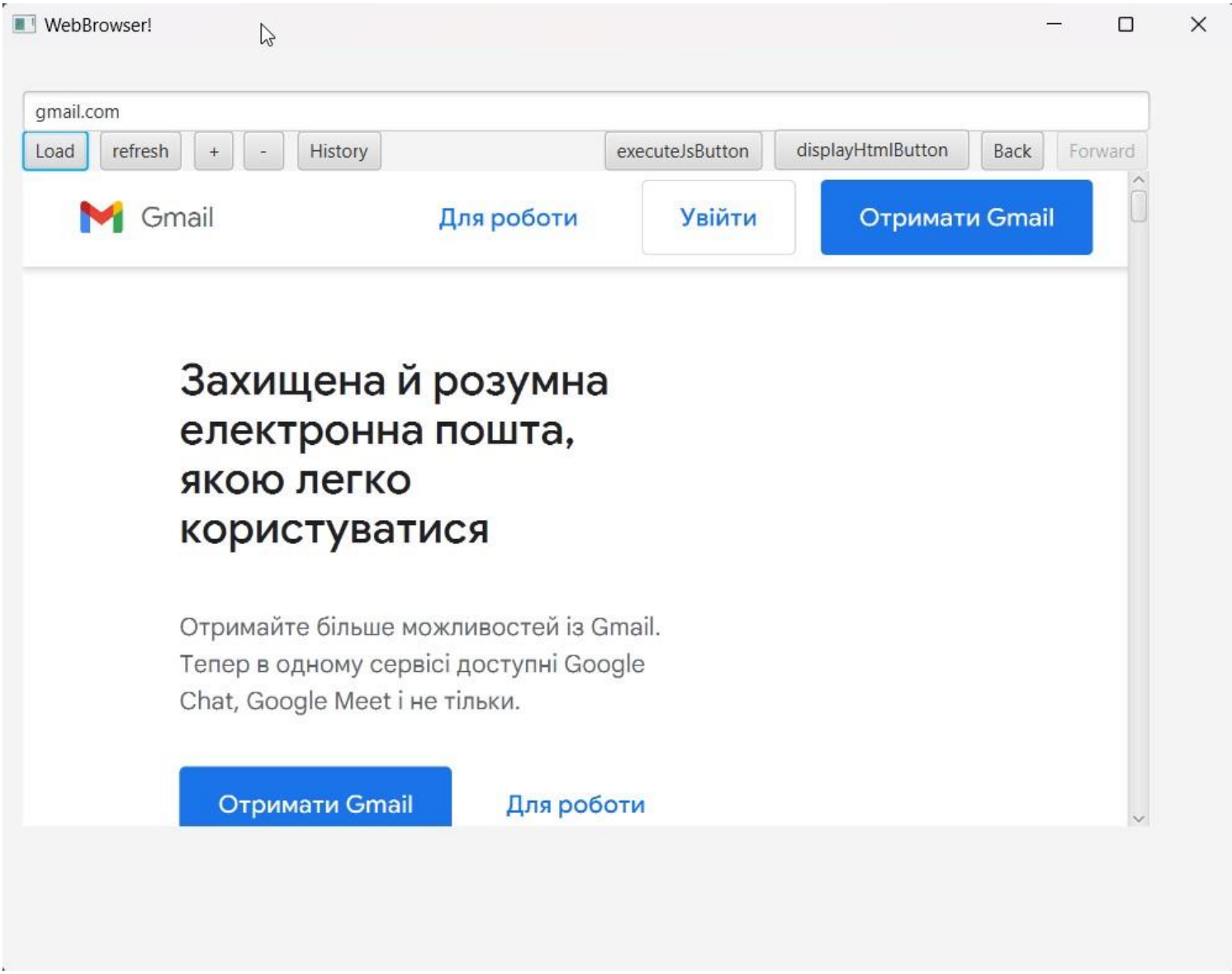
```
57      @Override
58      public void loadPage(String url) {
59          progressBar.setVisible(true);
60          realPageLoader.loadPage(url);
61
62          realPageLoader.getLoadWorkerStateProperty().addListener((observable, oldValue, newValue) -> {
63              if (newValue == Worker.State.SUCCEEDED) {
64                  progressBar.setVisible(false);
65                  handleResponse(realPageLoader.getEngine().getLocation());
66              }
67          });
68      }
69
        2 usages  Qwertua
70      @Override
71      public void reloadPage() {
72          progressBar.setVisible(true);
73          realPageLoader.reloadPage();
74
75          realPageLoader.getLoadWorkerStateProperty().addListener((observable, oldValue, newValue) -> {
76              if (newValue == Worker.State.SUCCEEDED) {
77                  progressBar.setVisible(false);
78                  handleResponse(realPageLoader.getEngine().getLocation());
79              }
80          });
81      }
82
        4 usages  Qwertua
83      @Override
84      public ReadOnlyObjectProperty<Worker.State> getLoadWorkerStateProperty() {
85          return realPageLoader.getLoadWorkerStateProperty();
86      }
```

```java
@Override
public void loadPage(String url) {
    /*Встановлюємо видимою шкалу завантаження сторінки*/
    progressBar.setVisible(true);
    /*завантаження сторінки  за допомогою realPageLoader*/
    realPageLoader.loadPage(url);
    /*відстежуємо зміну стану за допомогою слухача подій*/
    realPageLoader.getLoadWorkerStateProperty().addListener((observable, oldValue, newValue) -> {
        if (newValue == Worker.State.SUCCEEDED) {
            /*коли стан завантаження успішний прибираємо шкалу завантаження сторінки та викликаємо метод handleresponse*/
            progressBar.setVisible(false);
            handleResponse(realPageLoader.getEngine().getLocation());
        }
    });
}

2 usages  Qwertua *
@Override
public void reloadPage() {
    /*Встановлюємо видимою шкалу завантаження сторінки*/
    progressBar.setVisible(true);
    /*перезавантаження сторінки  за допомогою realPageLoader*/
    realPageLoader.reloadPage();
    /*відстежуємо зміну стану за допомогою слухача подій*/
    realPageLoader.getLoadWorkerStateProperty().addListener((observable, oldValue, newValue) -> {
        if (newValue == Worker.State.SUCCEEDED) {
            /*коли стан завантаження успішний прибираємо шкалу завантаження сторінки та викликаємо метод handleresponse*/
            progressBar.setVisible(false);
            handleResponse(realPageLoader.getEngine().getLocation());
        }
    });
}
```

```java
4 usages  Qwertua
@Override
/*Повертає властивість стану завантаження з RealPageLoader.*/
public ReadOnlyObjectProperty<Worker.State> getLoadWorkerStateProperty() {
    return realPageLoader.getLoadWorkerStateProperty();
}
```

Результат:


WebBrowser!
gmail.com
Load
refresh
+
-
History
executeJsButton
displayHtmlButton
Back
Forward
Gmail
Для роботи
Увійти
Отримати Gmail
Захищена й розумна електронна пошта, якою легко користуватися
Отримайте більше можливостей із Gmail. Тепер в одному сервісі доступні Google Chat, Google Meet і не тільки.
Отримати Gmail
Для роботи



Висновок: У даній лабораторній робот я детально розглянули та практично реалізували шаблон проектування «Proxy» відповідно до своєї теми.