ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ РЕАЛІЗАЦІЇ ШАБЛОНУ TEMPLATE METHOD

Мета

1 Закріпити інформацію про шаблони, отриману в результаті вивчення джерел інформації.

2 Набути навичок практичного використання шаблонів для вирішення прикладних задач.

Завдання роботи

Створити проект, який на основі даних колекції буде формувати звіт, який має:

– заголовок звіту (header), в якому подається назва звіту, дата його підготовки, період, за який формується звіт та інша інформація;

– дані звіту (data);

– «підвал» звіту (footer), в якому можуть бути представлені дані працівника, який готував звіт, місце для підпису та інша інформація.

Звіт необхідно створити двома способами:

– як текстовий файл (*.txt);

– як сторінку браузера (*.html).

Враховуючи, що перелік форматів файлів не включає всіх типів файлів, придатних для збереження даних, при розробці архітектури програми необхідно передбачити можливість розширення цього списку. У цьому випадку у якості рішення можливо використати шаблон проектування *Template Method*. Це поведінковий шаблон проектування, який визначає скелет алгоритму, перекладаючи відповідальність за деякі його етапи на підкласи. Шаблон дозволяє підкласам замінювати кроки алгоритму, не змінюючи його загальної структури.

Хід роботи

1 Ознайомлення з шаблоном проектування Template Method

1 Найменування: Template Method (шаблонний метод).

2 Клас шаблону:

– шаблонний метод — це поведінковий шаблон проектування (Behavioral Design Pattern), який визначає структуру алгоритму, надаючи можливість підкласам змінювати певні кроки алгоритму без зміни його структури.

3 Рівень використання:

– шаблонний метод використовується на рівні класів або компонентів;

– зазвичай він застосовується в рамках об'єктно-орієнтованого проектування для створення класів, які містять основну структуру алгоритму, але дозволяють підкласам змінювати частини цього алгоритму без необхідності змінювати саму структуру.

4 Умови використання:

– спільна основна структура алгоритму: шаблонний метод використовується, коли є алгоритм, що складається з декількох етапів, і цей алгоритм має однакову структуру у багатьох класах, але певні етапи можуть бути реалізовані по-різному;

– клас з загальною структурою: створюється базовий клас (або абстрактний клас), який містить основний алгоритм, цей клас має метод шаблону, який визначає кроки алгоритму, а деякі з кроків можуть бути абстрактними або надані підкласам для перевизначення;

– поліморфізм: підкласам надається можливість змінювати певні частини алгоритму без зміни його загальної структури;

– незмінність основної структури: алгоритм в цілому не змінюється під час виконання, а тільки окремі етапи змінюються у підкласах;

– часто використовується разом із іншими шаблонами проектування: наприклад, часто використовується разом з патерном Strategy або State, де підклас може змінювати поведінку на різних етапах алгоритму.

2 Розроблення класів для реалізації завдання

2.1 Реалізація шаблону Template Method

1 GenerateReportServlet (Додаток А):

– обробляє HTTP запити на генерацію звіту;

– отримує параметри з форми, такі як тип звіту та фільтр, і викликає метод для створення звіту;

– після генерації звіту виводиться відповідь на веб-сторінку, де користувач бачить результат — успіх або помилку;

– це реалізація шаблонного методу, оскільки вона визначає загальну структуру процесу створення звіту (отримання параметрів, виклик генератора звіту, виведення результату), залишаючи конкретну реалізацію формату звіту підкласам (наприклад, для тексту чи HTML).

2 OfficeWorkerListReportPreparer (Додаток Б):

– визначає шаблонний метод createReport;

– складається з отримання заголовка звіту (getHeader), даних для звіту (getData), отримання футера звіту (getFooter);

– ці кроки реалізуються в підкласах, що дозволяє змінювати лише специфічні частини звіту, не змінюючи його загальну структуру;

– містить метод для збереження звіту у файл (save).

3 HTMLReportPreparer (Додаток В):

– є підкласом OfficeWorkerListReportPreparer і реалізує специфічні кроки шаблону для HTML звітів;

– метод getHeader генерує HTML-код для заголовка звіту, getData створює таблицю з даними працівників, а getFooter додає підпис у форматі HTML.

4 MyReportPreparer (Додаток Г):

– конкретна реалізація шаблону, але для текстових звітів;

– реалізує методи для генерації текстового звіту, де дані форматуються як текст з використанням специфікацій форматування.

5 TXTReportCreator (Додаток Д):

– відповідає за виклик відповідного генератора звітів на основі типу звіту (HTML або TXT);

– завантажує типи звітів з файлу reports_types.dat і намагається створити інстанс відповідного класу підготовки звіту (наприклад, HTMLReportPreparer або MyReportPreparer);

– після цього викликається метод для створення звіту з передачею необхідних даних.

6 Officeworkers.jsp (Додаток Е):

– jsp-сторінка, на якій користувач може вибрати тип звіту (HTML або TXT) і ввести необов'язковий фільтр для звіту;

– форма відправляється на сервлет GenerateReportServlet, який обробляє запит і генерує звіт.

3 UML діаграма класів для розробленої імплементації

UML діаграма класів для розробленої імплементації зображена на рис. 1.

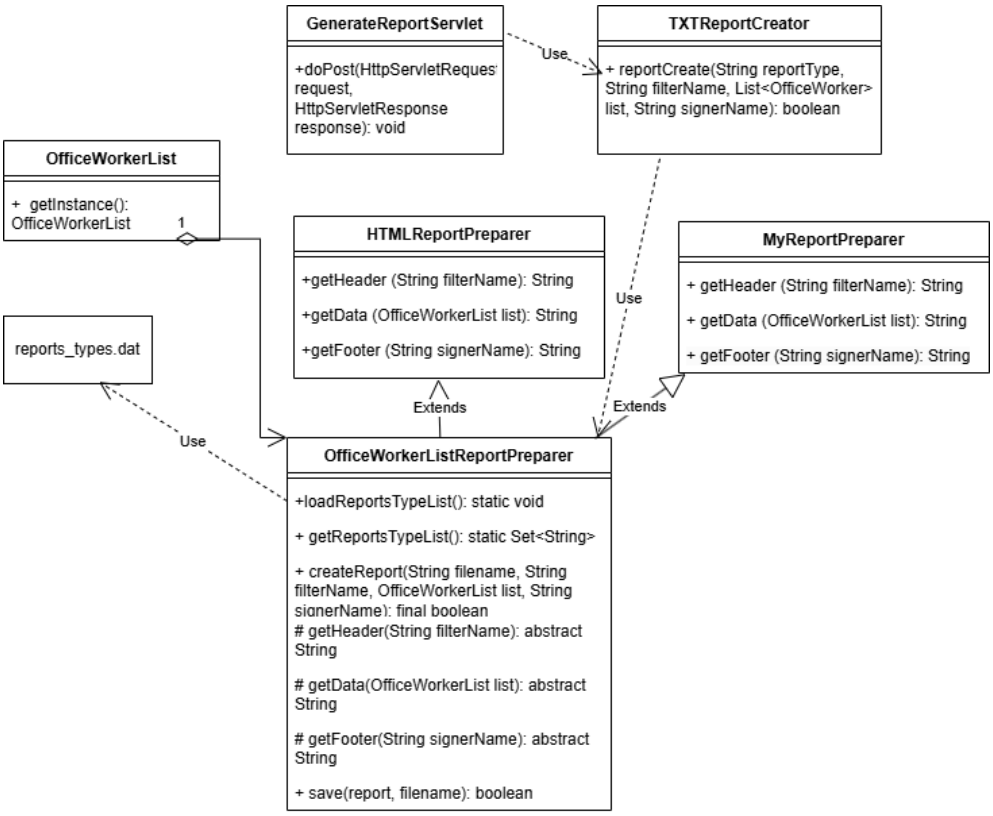

Рисунок 1 – UML діаграма класів для розробленої імплементації

5 Вигляд веб-сторінки

Інтерфейс стартової сторінки представлено на рисунку 2.

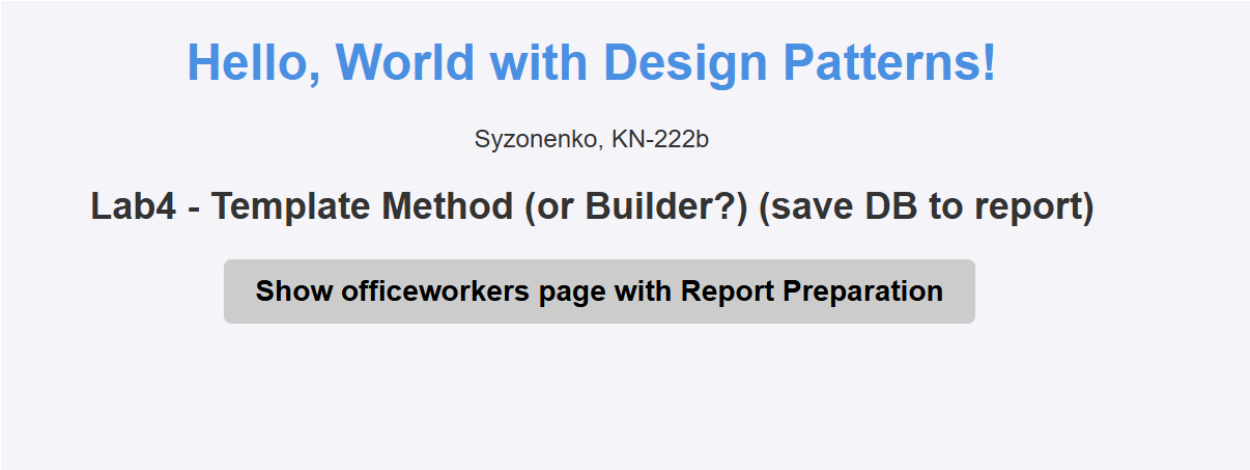

Рисунок 2 – Інтерфейс стартової сторінки

Інтерфейс сторінки з даними робітників представлено на рисунку 3.

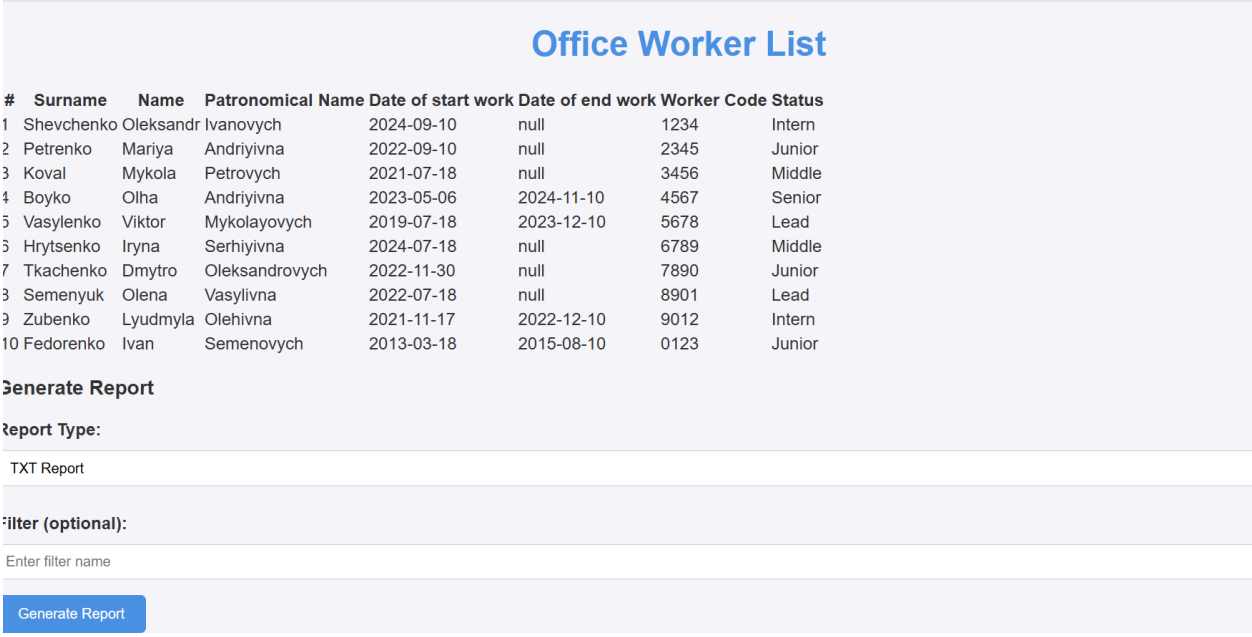

Рисунок 3 – Інтерфейс сторінки officeworkers.jsp

6 Вигляд звітів

Приклад звіту у форматі TXT зображено на рис. 4.

```
Generated by: Yelyzaveta Syzonenko

     # Surame                    Name WorkerCode Status
     1 Shevchenko          Oleksandr 1234       Intern
     2 Petrenko               Mariya 2345       Junior
     3 Koval                  Mykola 3456       Middle
     4 Boyko                    Olha 4567       Senior
     5 Vasylenko              Viktor 5678       Lead
     6 Hrytsenko               Iryna 6789       Middle
     7 Tkachenko              Dmytro 7890       Junior
     8 Semenyuk                Olena 8901       Lead
     9 Zubenko               Lyudmyla 9012       Intern
    10 Fedorenko                Ivan 0123       Junior

Prepared by Group: KN-222b. University: KhPI
```

Рисунок 4 – Файл TXT_officeworkers.txt

Приклад звіту у форматі html зображено на рис. 5.

```
<html><head><title>Office Worker Report</title></head><body><h1>Office Worker Report</h1><table border="1"><tr>
<th>#</th><th>Surname</th><th>Name</th><th>Worker Code</th><th>Status</th></tr><tr><td>1</td><td>Shevchenko</td><td>
Oleksandr</td><td>1234</td><td>Intern</td></tr><tr><td>2</td><td>Petrenko</td><td>Mariya</td><td>2345</td><td>
Junior</td></tr><tr><td>3</td><td>Koval</td><td>Mykola</td><td>3456</td><td>Middle</td></tr><tr><td>4</td><td>
Boyko</td><td>Olha</td><td>4567</td><td>Senior</td></tr><tr><td>5</td><td>Vasylenko</td><td>Viktor</td><td>5678</td>
<td>Lead</td></tr><tr><td>6</td><td>Hrytsenko</td><td>Iryna</td><td>6789</td><td>Middle</td></tr><tr><td>7</td><td>
Tkachenko</td><td>Dmytro</td><td>7890</td><td>Junior</td></tr><tr><td>8</td><td>Semenyuk</td><td>Olena</td>
<td>8901</td><td>Lead</td></tr><tr><td>9</td><td>Zubenko</td><td>Lyudmyla</td><td>9012</td><td>Intern</td></tr><tr>
<td>10</td><td>Fedorenko</td><td>Ivan</td><td>0123</td><td>Junior</td></tr></table><p>Prepared by Group: KN-222b.
University: KhPI</p></body></html>
```

Рисунок 5 – Файл html_officeworkers.html

Висновки

У процесі виконання цієї лабораторної роботи було досліджено шаблон проектування Template Method, який є важливим інструментом для організації кроків алгоритму, дозволяючи визначити загальний процес у базовому класі, а специфічні кроки — в підкласах. Це дозволяє забезпечити зручну та гнучку реалізацію алгоритмів, де певні етапи можуть бути змінені або розширені, зберігаючи при цьому загальну структуру.

Робота з шаблоном Template Method дозволила краще зрозуміти, як організовувати програму так, щоб основна логіка алгоритму була захищена від

змін, а специфічні частини могли бути легко адаптовані під конкретні вимоги. Завдяки цьому, шаблон забезпечує гнучкість у програмуванні, дозволяючи мінімізувати дублювання коду і забезпечити його легку модифікацію.

Порівнюючи шаблон Template Method з іншими шаблонами проектування, можна відзначити його перевагу в ситуаціях, коли необхідно забезпечити певну структуру роботи алгоритму з можливістю змінювати лише частини його реалізації. Це дозволяє зберегти єдність процесу, зберігаючи високу модульність і спрощуючи підтримку в майбутньому. Однак, неправильне використання шаблону може призвести до складності в розумінні та обслуговуванні коду, якщо надто велика частина алгоритму реалізована в базовому класі.

Вивчення шаблону Template Method дозволило зрозуміти важливість визначення базових кроків алгоритму в одному місці, при цьому дозволяючи підкласам модифікувати лише необхідні частини без порушення загальної структури. Це знання є корисним для створення програм, що мають чітко визначені алгоритми, але при цьому потребують можливості адаптації під конкретні умови чи вимоги. Отриманий досвід дозволяє зробити висновок, що шаблон Template Method є потужним інструментом для організації алгоритмів, який дозволяє створювати гнучкі та розширювані системи.

ДОДАТОК А

Код класу GenerateReportServlet

```
package stusyo222b.appz_4.template;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import stusyo222b.appz_4.entities.OfficeWorker;
import stusyo222b.appz_4.entities.OfficeWorkerList;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.List;
import java.util.stream.Collectors;

@WebServlet("/GenerateReportServlet")
public class GenerateReportServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        // Get parameters from the form
        String reportType = request.getParameter("reportType");
        String filterName = request.getParameter("filterName") != null ? request.getParameter("filterName") : "";
        String signerName = "Group: KN-222b. University: KhPI";


        boolean result = TXTReportCreator.reportCreate(reportType, filterName, OfficeWorkerList.getInstance(),
signerName);

        // Prepare the response
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><head><title>Report Generation</title></head><body>");
        if (result) {
            out.println("<h2>Report created successfully!</h2>");
        } else {
            out.println("<h2>Error: Failed to create the report.</h2>");
        }
        out.println("<a href=\"officeworkers.jsp\">Back to OfficeWorkers</a>");
        out.println("</body></html>");
        out.close();
    }
}
```

ДОДАТОК Б

Код класу OfficeWorkerListReportPreparer

```
package stusyo222b.appz_4.template;


import stusyo222b.appz_4.entities.OfficeWorkerList;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.util.*;

public abstract class OfficeWorkerListReportPreparer {

    static protected Map<String, String> reportsTypes = null;
    public static final String workPath =
            Thread.currentThread().getContextClassLoader().getResource("").getPath();

    public static void loadReportsTypeList() {
        boolean flOK = false;
        String fullPath = workPath+"reports_types.dat";

        Path path = Paths.get(fullPath.substring(1));
        try {
            List<String> typesList = Files.readAllLines(path, StandardCharsets.UTF_8);
            if (typesList.isEmpty()) {
                throw new RuntimeException("File with reports type was found empty!");
            }
            reportsTypes = new HashMap<>();
            for (String s : typesList) {
                reportsTypes.put(s.split("=")[0], s.split("=")[1]);
            }
            flOK = true;
        } catch (IOException e) {
            System.err.println("Problems with file...");
            flOK = false;
        } catch (Exception e) {
            System.err.println("File with reports types opened with problems...");
            flOK = false;
        }

    }

    public static Set<String> getReportsTypeList() {
        return reportsTypes.keySet();
    }

    public final boolean createReport(String filename, String filterName, OfficeWorkerList list, String signerName) {
        StringBuilder sb = new StringBuilder();
        sb.append(getHeader(filterName))
                .append(getData(list))
                .append(getFooter(signerName));
        String report = sb.toString();
        return save(report, filename);
```

```
    }

    abstract String getHeader(String filterName);
    abstract String getData(OfficeWorkerList list);
    abstract String getFooter(String signerName);

    public static boolean save(String report, String filename) {
        boolean flOK = false;
        if (!report.isEmpty()) {
            List<String> linesOfReport = Arrays.stream(report.split(System.lineSeparator())).toList();
            Path filepath = Paths.get(filename);
            try {
                Files.write(filepath, linesOfReport, StandardCharsets.UTF_8, StandardOpenOption.CREATE,
StandardOpenOption.TRUNCATE_EXISTING, StandardOpenOption.WRITE);
                flOK = true;
            } catch (IOException e) {
                flOK = false;
                System.err.println("Problem with filename \""+filename+"\"");
            }
        } else {
            System.err.println("List is null");
            flOK = false;
        }
        return flOK;
    }

}
```

ДОДАТОК В

Код HTMLReportPreparer

```
package stusyo222b.appz_4.template;


import stusyo222b.appz_4.entities.OfficeWorkerList;

public class HTMLReportPreparer extends OfficeWorkerListReportPreparer {

    @Override
    String getHeader(String filterName) {
        StringBuilder header = new StringBuilder();
        header.append("<html><head><title>Office Worker Report</title></head><body>")
                .append("<h1>Office Worker Report</h1>");
        if (!filterName.isEmpty()) {
            header.append("<h2>Filter: ").append(filterName).append("</h2>");
        }
        header.append("<table border=\"1\"><tr>")
                .append("<th>#</th><th>Surname</th><th>Name</th><th>Worker Code</th><th>Status</th>")
                .append("</tr>");
        return header.toString();
    }

    @Override
    String getData(OfficeWorkerList list) {
        StringBuilder data = new StringBuilder();
        if (!list.isEmpty()) {
            for (int number = 0; number < list.size(); number++) {
                data.append("<tr>")
                        .append("<td>").append(number + 1).append("</td>")
                        .append("<td>").append(list.get(number).getSurname()).append("</td>")
                        .append("<td>").append(list.get(number).getName()).append("</td>")
                        .append("<td>").append(list.get(number).getWorkerCod()).append("</td>")

.append("<td>").append(list.get(number).getOfficeWorkerStatus().getDisplayName()).append("</td>")
                        .append("</tr>");
            }
        } else {
            data.append("<tr><td colspan=\"4\">No data available</td></tr>");
        }
        return data.toString();
    }

    @Override
    String getFooter(String signerName) {
        return "</table><p>Prepared by " + signerName + "</p></body></html>";
    }
}
```

ДОДАТОК Г

Код класу MyReportPreparer

```
package stusyo222b.appz_4.template;

import stusyo222b.appz_4.entities.OfficeWorkerList;

public class MyReportPreparer extends OfficeWorkerListReportPreparer {


   @Override
   String getHeader(String filterName) {
      StringBuilder h = new StringBuilder(System.lineSeparator()+"Generated by: Yelyzaveta Syzonenko"
            +System.lineSeparator()+System.lineSeparator());
      if (!filterName.isEmpty()) {
         h.append(filterName).append(System.lineSeparator());
      }
      return h.toString();
   }

   @Override
   String getData(OfficeWorkerList list) {
      String spec = "%6s %-20s %10s %-10s %-20s"+System.lineSeparator();
      String specD = "%6s %-20s %10.2f %-10s %-20s"+System.lineSeparator();
      StringBuilder data = new StringBuilder(String.format(spec,"#","Surame","Name","WorkerCode", "Status"));
      if (!list.isEmpty()) {
         for (int number = 0;number < list.size();number++) {
            data.append(String.format(spec,number+1,list.get(number).getSurname(), list.get(number).getName(),
                  list.get(number).getWorkerCod(), list.get(number).getOfficeWorkerStatus().getDisplayName()));
         }
         data.append(System.lineSeparator());
      }
      return data.toString();
   }

   @Override
   String getFooter(String signerName) {
      return "Prepared by "+ signerName;
   }
}
```

ДОДАТОК Д

Код класу TXTReportCreator

```
package stusyo222b.appz_4.template;

import stusyo222b.appz_4.entities.OfficeWorkerList;

import static stusyo222b.appz_4.template.OfficeWorkerListReportPreparer.reportsTypes;

public class TXTReportCreator {

   public static boolean reportCreate(String reportType, String filterName, OfficeWorkerList list, String
signerName) {
      boolean flOK = false;
      //For TXT
      OfficeWorkerListReportPreparer.loadReportsTypeList();
      if (reportsTypes.size() != 0) {
         OfficeWorkerListReportPreparer elrp = null;
         try {

            String typeReportPrefix = reportsTypes.get(reportType);
            if (typeReportPrefix!=null) {
               System.out.println("Type report prefix: " + typeReportPrefix);
               String reportPreparerName = "stusyo222b.appz_4.template." + typeReportPrefix+"ReportPreparer";
               System.out.println("Attempting to load class: " + reportPreparerName);
               elrp = (OfficeWorkerListReportPreparer) Class.forName(reportPreparerName).newInstance();
               System.out.println("Class loaded and instance created successfully.");
               String filename = reportType + "_officeworkers.txt";
               String fullname = OfficeWorkerListReportPreparer.workPath.substring(1) + filename;
               flOK = elrp.createReport(fullname, filterName, list, signerName);
               System.out.println("Saving file to: " + fullname);


            } else {
               System.err.println("No mapping found for reportType: " + reportType);
               flOK = false;
               System.err.println("\""+reportType+"\' <== Processing such type report not implemented...");
            }
         } catch (InstantiationException | IllegalAccessException | ClassNotFoundException e) {
            flOK = false;
            System.err.println("Problem when create OfficeWorkerListReportPreparer");
         }
      } else {
         flOK = false;
         System.err.println("List of SaverToFile empty");
      }
      return flOK;
   }
}
```

ДОДАТОК Е

Код сторінки officeworkers.jsp

```
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page import="stusyo222b.appz_4.entities.OfficeWorkerList" %>
<%@ page import="stusyo222b.appz_4.entities.OfficeWorker" %>
<%@ page import="java.util.List" %>
<%@ page import="stusyo222b.appz_4.template.OfficeWorkerListReportPreparer" %>
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Office Workers</title>
  <style type="text/css">
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f9;
      color: #333;
      margin: 0;
      padding: 0;
    }
    .content {
      max-width: 800px;
      margin: 40px auto;
      padding: 20px;
      background-color: #fff;
      border-radius: 8px;
      box-shadow: 0 0 15px rgba(0, 0, 0, 0.1);
    }
    h1 {
      color: #4a90e2;
      text-align: center;
    }
    label {
      font-weight: bold;
      display: block;
      margin: 10px 0 5px;
    }
    input[type="text"],
    input[type="date"],
    select {
      width: 100%;
      padding: 8px;
      margin: 5px 0 15px;
      border: 1px solid #ccc;
      border-radius: 4px;
      box-sizing: border-box;
    }
    .error-message {
      color: red;
      font-weight: bold;
      text-align: center;
      margin-bottom: 20px;
    }
    button {
      background-color: #4a90e2;
      color: white;
      padding: 10px 20px;
```

```
    border: none;
    border-radius: 5px;
    cursor: pointer;
   }
   button:hover {
    background-color: #357ab7;
   }
   a {
    display: inline-block;
    padding: 10px 20px;
    margin-left: 10px;
    background-color: #ccc;
    color: black;
    text-decoration: none;
    border-radius: 5px;
   }
   a:hover {
    background-color: #aaa;
   }
   @media (max-width: 600px) {
    .content {
     padding: 15px;
     box-shadow: none;
    }
    h1 {
     font-size: 1.5em;
    }
   }
  </style>
</head>
<body>
<h1>Office Worker List</h1>

<table>
 <thead>
 <tr>
  <th>#</th>
  <th>Surname</th>
  <th>Name</th>
  <th>Patronomical Name</th>
  <th>Date of start work</th>
  <th>Date of end work</th>
  <th>Worker Code</th>
  <th>Status</th>
 </tr>
 </thead>
 <tbody>
 <%
  OfficeWorkerList officeworkers = OfficeWorkerList.getInstance();
  int counter = 1;
  for (OfficeWorker off : officeworkers) {
 %>
 <tr>
  <td><%= counter++ %></td>
  <td><%= off.getSurname() %></td>
  <td><%= off.getName() %></td>
  <td><%= off.getPname() %></td>
  <td><%= off.getStartWork() %></td>
  <td><%= off.getEndWork() %></td>
  <td><%= off.getWorkerCod() %></td>
  <td><%= off.getOfficeWorkerStatus().getDisplayName() %></td>
 </tr>
 <%
```

```
    }
  %>
  </tbody>
</table>

<div class="form-container">
  <h3>Generate Report</h3>
  <form action="GenerateReportServlet" method="post">
    <label for="reportType">Report Type:</label>
    <select name="reportType" id="reportType">
      <%
        OfficeWorkerListReportPreparer.loadReportsTypeList();
        for (String reportType : OfficeWorkerListReportPreparer.getReportsTypeList()) {
      %>
      <option value="<%= reportType %>"><%= reportType.toUpperCase() %> Report</option>
      <%
        }
      %>
    </select>
    <br>
    <label for="filterName">Filter (optional):</label>
    <input type="text" name="filterName" id="filterName" placeholder="Enter filter name">
    <br>
    <button type="submit">Generate Report</button>
  </form>

</div>
</body>
</html>
```