HW3 ASSIGNMENT

**SUMMARY:**

***HtmlGenerator.java:***

**Explanation:** This class is designed to generate HTML code for displaying a course history table. It defines several methods, including ‘generateHTML’, which constructs the entire HTML document, including title, styling, header, and course data rows. The ‘headerRow’ method generates the table headers for “Semester” and “Course”, while the ‘courseRows’ method dynamically creates data rows based on provided semesters and course names, allowing for customization for different academic terms. The ‘createTableRow’ method is responsible for constructing individual table rows for each course. Inline CSS styling is included for consistent formatting, and the resulting HTML can be easily integrated into web applications or saved to files for display in a web browser.

**Dry Principles Applied:**

1. **Reusable HTML Building Blocks:** The class has methods for creating header rows, data rows, and the whole HTML structure ('generateHTML'). You can eliminate duplication and improve the code's maintainability by encapsulating the logic involved in creating HTML in methods.
2. **Centralized Styling:** The inline CSS styles are centralized and are defined in the'style' method of the 'head' section. This makes sure that stylistic rules may be applied uniformly to different HTML components and are defined just once. Instead of doing it across the code, you may alter the styling in one location.
3. **Parameterized Logic:** The 'courseRows' method takes two input parameters as inputs (semesters and courseNames), and it uses those parameters to construct data rows dynamically based on the inputted information. By using conditional logic to choose courses according to the semester, it avoids writing the same code for every semester. The code may adapt to various datasets using this parameterized method without using redundant logic.
4. **Consistent Naming Conventions:** Variable and method names in the class are consistent and meaningful. Because of this uniformity, fewer unnecessary comments are required to describe the function of the variables and the methods. It adheres to the DRY principle, which aims to make code self-explanatory, by following the convention of using descriptive names.

***HtmlFileWriter.java:***

**Explanation:** The ‘HtmlFileWriter’ class provides a method, ‘writeHTMLToFile’, for writing HTML content to a specified file. Within the method, a ‘FileWriter’ is created to handle file writing, and a ‘PrintWriter’ is used to write formatted text to the ‘FileWriter’. The HTML content, provided as a ‘ContainerTag’ object named ‘html’, is written to the file using the ‘renderFormatted()’ method. After writing, the ‘PrintWriter’ and ‘FileWriter’ are closed to release resources. If an ‘IOException’ occurs during the file writing process, an error message is printed, indicating the issue. Conversely, a success message is printed if the HTML file is generated without errors, including the name of the file that was successfully created. Overall, this class encapsulates the file writing logic for HTML content, making it convenient to generate and save HTML files.

**Dry Principles Applied:**

1. **Modularization:** The class's single method, writeHTMLToFile, contains the logic for writing files. Writing HTML text to a file is the sole purpose of this technique. The code is modular and adheres to the DRY principle by isolating this functionality so that it may be reused without being duplicated.
2. **Code Reuse:** The class leverages the ‘ContainerTag’ class from the j2html library to handle HTML rendering. Instead of duplicating the rendering code throughout the application, the ‘writeHTMLToFile’ method accepts a ‘ContainerTag’ as a parameter and uses it to generate HTML content. This approach promotes code reuse and adheres to the DRY principle by centralizing HTML rendering.

***HtmlFileOpener.java:***

**Explanation:** The ‘HtmlFileOpener’ class provides a method, ‘openHTMLFile’, for opening an HTML file in a web browser. The method first checks if the specified HTML file exists by creating a ‘File’ object representing the file and verifying its existence. It then determines the operating system using ‘System.getProperty("os.name")’ and checks if it's Windows by examining the OS name in a case-insensitive manner. If the OS is Windows, it employs a ‘ProcessBuilder’ to execute a command that opens the HTML file in the default web browser, setting the working directory to the current user's directory. In case the operating system is not Windows, it prints an error message indicating that opening in a web browser is not supported on the platform. If the HTML file doesn't exist, it prints an error message, and if an ‘IOException’ occurs during the process, it prints an error message with the exception details.

**Dry Principles Applied:**

1. **Modularization and Single Responsibility: ‘**openHTMLFile’, a method of the class, contains the logic for opening a file. Opening an HTML file in a web browser is the only task this technique must complete. The code is modular and adheres to the DRY principle because this functionality is isolated. By doing this, it prevents the application from using the same logic several times to access HTML files.

**SCREENSHOTS:**

1. **Application up and running:**

**A black rectangle with white text

Description automatically generated**

1. **Table Generation:**

**A list of courses

Description automatically generated**